

AD-A150 162

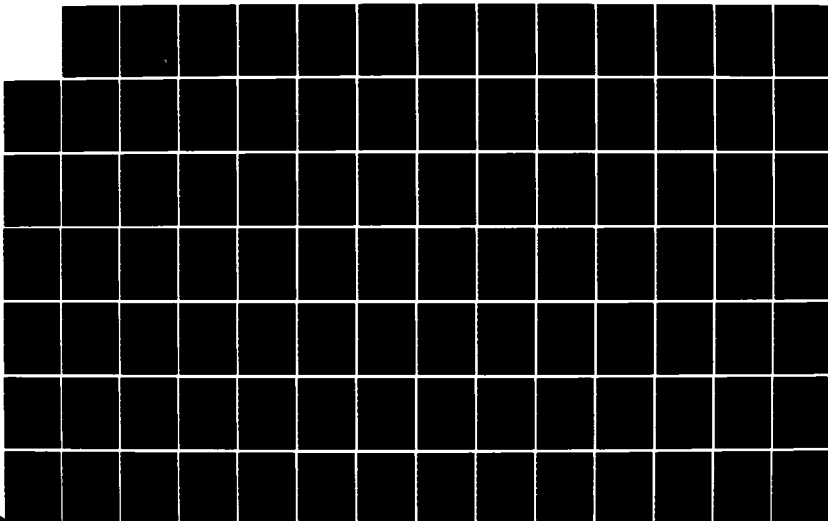
COMPUTER CENTER CDC REFERENCE MANUAL(U) DAVID W TAYLOR
NAVAL SHIP RESEARCH AND DEVELOPMENT CENTER BET..
D V SOMMER ET AL. SEP 84 DTNSRDC/CHLD-84-10

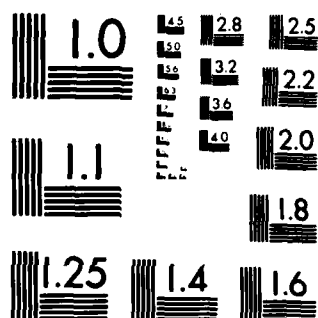
1/3

UNCLASSIFIED

F/G 9/2

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

CMLD-84-10

AD-A150 162

COMPUTER CENTER CDC REFERENCE MANUAL

DAVID W. TAYLOR NAVAL SHIP RESEARCH AND DEVELOPMENT CENTER

Bethesda, Maryland 20084



COMPUTER CENTER
CDC
REFERENCE MANUAL

by

David V. Semmer
Sharon E. Good

APPROVED FOR PUBLIC RELEASE: DISTRIBUTION UNLIMITED

Computation, Mathematics and Logistics Department
Departmental Report

DTIC FILE COPY

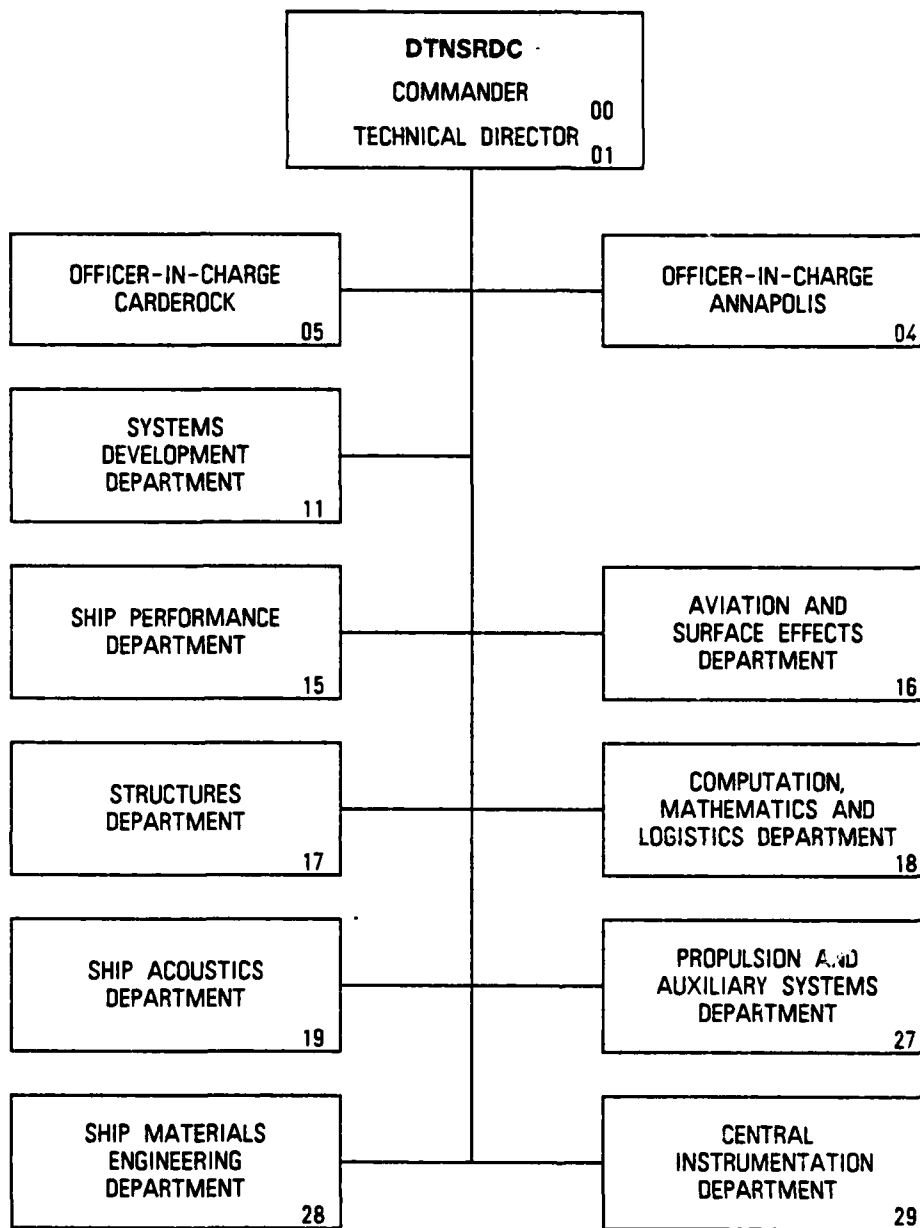
DTIC
ELECTE
JAN 18 1985
E

September 1984

CMLD-84-10

85 01 11 080

MAJOR DTNSRDC ORGANIZATIONAL COMPONENTS



REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM												
1. REPORT NUMBER CMLD-84-10	2. GOVT ACCESSION NO. AD-A150162	3. RECIPIENT'S CATALOG NUMBER												
4. TITLE (and Subtitle) Computer Center CDC Reference Manual		5. TYPE OF REPORT & PERIOD COVERED Final												
7. AUTHOR(s) David V. Sommer Sharon E. Good		6. PERFORMING ORG. REPORT NUMBER												
9. PERFORMING ORGANIZATION NAME AND ADDRESS DTNSRDC, User Services, Code 1892 Bethesda, Maryland 20084		8. CONTRACT OR GRANT NUMBER(s)												
11. CONTROLLING OFFICE NAME AND ADDRESS Computation, Mathematics & Logistics Dept. Computer Facilities Division (189)		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS												
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		12. REPORT DATE September 1984												
		13. NUMBER OF PAGES 392												
		15. SECURITY CLASS. (of this report) Unclassified												
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE												
16. DISTRIBUTION STATEMENT (of this Report) Approved for Public Release; Distribution Unlimited														
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)														
18. SUPPLEMENTARY NOTES														
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) <table border="0"> <tr> <td>CDC CYBER 170</td> <td>Hardware</td> <td>Remote Job Entry</td> </tr> <tr> <td>Computer</td> <td>Interactive Processing</td> <td>Software</td> </tr> <tr> <td>Control Statement</td> <td>NOS/BE Operating System</td> <td>Documentation</td> </tr> <tr> <td>Graphics</td> <td>Programming languages</td> <td></td> </tr> </table>			CDC CYBER 170	Hardware	Remote Job Entry	Computer	Interactive Processing	Software	Control Statement	NOS/BE Operating System	Documentation	Graphics	Programming languages	
CDC CYBER 170	Hardware	Remote Job Entry												
Computer	Interactive Processing	Software												
Control Statement	NOS/BE Operating System	Documentation												
Graphics	Programming languages													
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) <p>The Computer Center CDC Reference Manual provides an introduction to the CDC NOS/BE Operating System for applications programmers. Some information has been distilled from many individual documents and augmented to reflect usage at DTNSRDC. Control statement examples and descriptions of hardware and software are included.</p>														

David W. Taylor
Naval Ship Research and Development Center
Bethesda, Maryland 20084

*
*
* Computer Center *
* CDC Reference Manual *
*
*

by
David V Sommer
Sharon E Good
User Services Branch

Code 1892

	Carderock	Annapolis
Phone	(202) 227-1907	(301) 267-3343
Autovon	287-1907	281-3343

Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A-1	

For recorded message on computer status (202) 227-3043

Questions and requests for more detailed information
should be directed to Code 1892, Bldg. 17, Rm. 100.



Computation, Mathematics and Logistics Department
Departmental Report

September 1984

CMLD-84-10

..

Through Revision 0 (Sept 1984)

..

84/09/30

Rev0

CDC CYBER

Page R-1

*** Revision Record ***

Revision	Description
0 (Sep 84)	Original printing.

*** Abbreviations ***

Throughout this manual, the following abbreviations will be used to indicate reference manuals:

abbreviation	manual
AAMUG	Advanced Access Methods V2 User's Guide
BAMUG	Basic Access Methods V1.5 User's Guide
CCIRM	Computer Center Introductory Reference Manual
CCRM	Computer Center CDC Reference Manual (this manual)
CLIB	Computer Center Libraries
CLIB/M	CLIB/MNSRDC (programs)
CLIB/N	CLIB/NSRDC and NSRDC5 (subprograms)
CLIB/P	CLIB/PROCFIL (procedures)
CLIB/U	CLIB/UTILITY (programs)
COBOL4	COBOL Version 4 Reference Manual
COBOL5	COBOL Version 5 Reference Manual
CONV	3.3 to 3.4 Conversion Aids
DEBUG	FORTRAN Extended Debug User's Guide
FORM	FORM Reference Manual
FTN4	FORTRAN Extended V4 Reference Manual
FTN5	FORTRAN V5 Reference Manual
INT	Intercom V4 Reference Manual
INTCOB	Intercom for COBOL Users
INTFTN	Intercom for FORTRAN Users
LOADER	Loader Reference Manual
NOSBE	NOS/BE 1 Reference Manual
POLICY	Computer Center Policy
RMBAM	Record Manager Basic Access Methods 1 Ref Manual
RMUG	Record Manager User's Guide
SORT4	Sort/Merge Version 4 Reference Manual
SORT5	Sort/Merge Version 5 Reference Manual
SUG	SCOPE 3.4 User's Guide
UPDATE	UPDATE Reference Manual

Many of these manuals are in the Reference Centers. See Appendix H for Reference Center locations and Appendix I for the form numbers of these and other manuals.

*** Page Numbering Scheme ***

Each chapter in this manual is numbered consecutively starting with page 1, except:

- a) After publication, as additional pages are added, those which are added to the end of a chapter will continue the numbering scheme; those inserted in the middle of a chapter will have .1, .2, etc., added to the page number they are to follow (e.g., 7-6.1 will follow 7-6).

*** Notation ***

Throughout the manual, when a word or phrase is enclosed in broken brackets (e.g., <list>), the <> are not part of the string.

*** Revisions ***

This printing is a major revision of the May 1981 Computer Center CDC Reference Manual (CMLD-81-21). It includes information on new hardware and is a complete restructuring of the manual. All pages with a date (at the top of the page) after May 1983 are either new or revised.

Additional or correction pages will be issued at intervals.

Table of Contents

Preface	
Abbreviations	i
Page Numbering Scheme	ii
Notation	ii
Revisions	ii
1 INTRODUCTION TO CDC COMPUTERS	1-1
Hardware Configuration	1-3
NOS/BE 1.5	1-5
2 USER INTERFACE WITH THE COMPUTER CENTER	2-1
General Information	2-1
Trouble Forms	2-1
Gripes	2-1
Refunds	2-2
ADP Control Center	2-2
Central Site EAM Processing	2-3
Punched Cards from Central Site	2-3
3 BATCH PROCESSING	3-1
Job Processing	3-2
Remote Batch Processing	3-3
Medium-speed Terminals	3-3
Medium-speed Remote Batch Processing Commands	3-4
Printer	3-4
Card Reader	3-5
Printer and Card Reader	3-5
Queue and Status Displays	3-5
Manipulating Queue-resident Files	3-6
Some Error Messages	3-7
Block Time	3-8
4 INTERACTIVE PROCESSING	4-1
Introduction	4-1
Sign-on Procedures	4-2
User Turnkey Password	4-3
SECURE/RESUME a Terminal	4-4
Sign-off Procedures	4-4
System Bulletin	4-4
Correcting and Interrupting	4-5
Intercom User Limitations	4-6

Program Construction - NETED	4-7
CDC Intercom EDITOR	4-9
Leaving and Re-entering EDITOR	4-10
Batch Runs from Interactive Terminals	4-11
Creation of Control Statement Record	4-11
Terminal Access to Batch Job Output	4-12
Send Job to a Printer After Terminal Access	4-12
Removing Jobs in Your Queues	4-12
Locating Jobs in the Queues	4-13
MYQ Command	4-13
FIND (or J) Command	4-14
Examples	4-14
Error Messages	4-14
Q Command	4-15
Examples	4-15
Messages Between Consoles	4-16
From Central Site Operator	4-16
To Central Site Operator	4-17
To Another Terminal	4-17
To a User Who is not Logged In	4-17
Saving Files	4-18
STORE, FETCH, DISCARD Commands	4-18
Additional Intercom Commands	4-19
XEQ Command	4-21
XEQ Examples	4-22
Page	4-23
Calculator Mode	4-26
ASCII and Intercom	4-27
Intercom Backup Decks	4-28
External Storage Media	4-29
Paper Tape -- Cassette	4-29
 5 JOB PROCESSING CONTROL STATEMENTS	 5-1
Introduction	5-1
Control Statement Formats	5-2
Job Card	5-3
CHARGE Card	5-5
WARNING	5-6
Other NOS/BE Control Statements	5-7
Describing a File	5-11
ROUTE	5-12
Special-forms Codes	5-15
ROUTE Examples	5-16

6	PROCEDURES (CCL)	6-1
	Introduction	6-1
	Interactive Execution	6-2
	Public-access Procedures - PROCFIL	6-2
	AUDIT	6-3
	MSAUDIT	6-3
	DOCGET	6-4
	GRIPE	6-5
	RENAMAC	6-6
	UTILITY	6-6
	XEROX	6-7
7	SECURITY	7-1
	Privacy Act of 1974	7-2
	Classified Microfiche	7-3
	Processing Classified Jobs	7-4
	Physical Security	7-4
	Permanent Files	7-4
	Job Card	7-5
	Device Set Password	7-6
8	LOADER	8-1
	Introduction	8-1
	Types of Loading	8-1
	Loader Control Statements	8-4
	LDSET	8-6
	Loader-related Control Statements	8-8
	Segmentation	8-10
	Segmentation Cautions	8-12
	Overlay	8-13
	Setting Up Overlay	8-14
	Sample Overlay	8-15
	Overlay Capsules	8-16
	Setting Up Capsules	8-17
	Compile, Load and Catalog Absolute Program	8-19
9	DISK FILES	9-1
	Introduction	9-1
	Record Manager	9-1
	File Organizer and Record Manager (FORM)	9-1
	Characteristics of Rotating Mass Storage Devices	9-2
	Permanent Files	9-3
	Permanent File Hints	9-3
	Automatic Permanent File Purge	9-4
	User Audit	9-5

Permanent File Commands	9-6
Hints on Permanent File Usage	9-8
Permanent File User Return Codes	9-9
Permanent File Examples	9-10
Device Sets	9-13
Device Set Control Statements	9-14
Device Set Utilities	9-15
Indexed Sequential	9-16
10 MASS STORAGE SYSTEM (MSS) FILES	10-1
Mass Storage System (MSS)	10-1
MSS Security	10-1
MSS File Purge	10-2
MSS for File Backup	10-2
MSS Commands	10-3
MSS Command Parameters	10-5
11 TAPE FILES	11-1
Tape Characteristics	11-1
SI Tapes	11-1
Stranger Tapes	11-2
7-track and 9-track Tapes	11-2
FORTRAN Tapes	11-3
Considerations for Creating Tapes for Data	
Interchange	11-3
Estimating Tape Requirements	11-4
Magnetic Tape Formats	11-7
Tape Assignment	11-8
Tape Care and Cleaning	11-9
VSN	11-10
Labelled Tapes	11-12
Unlabelled Tapes	11-14
IBM-style Multi-file Labelled Tapes	11-15
12 UTILITIES	12-1
NOS/BE Utilities	12-1
COPYBF/COPYCF/COPYBR/COPYCR	12-1
SKIPF/SKIPB	12-2
COPYL/COPYLM	12-2
COPYN	12-3
ITEMIZE	12-4
BKSP/COMBINE/COMPARE/COPY	12-5
Other Utilities	12-6
COPYE	12-6
COPYF/COPYR	12-7
COPYRM	12-8
COPYSF(COPYSBF)/COPYSR	12-9
COMPAR	12-10
Sample Utility Setups	12-11

13	FORTTRAN EXTENDED	13-1
	Introduction	13-1
	FTN5 Control Statement Parameters	13-2
	FTN5 PROGRAM Statement	13-7
	Object Program Execution	13-8
	File Name Replacement	13-8
	Print Limit Specification	13-8
	Post Mortem Dump Parameters	13-9
	User Parameters	13-10
	Accessing User Parameters	13-10
	FTN4 vs FTN5	13-11
	New Features of FTN5	13-15
	FORTTRAN, Version 5, Considerations	13-16
	FORTTRAN, Version 5, Hints	13-18
	FTN4 to FTN5 Conversion Aid	13-19
	Error Messages	13-21
	FORTRAN Compilation Errors	13-21
	FORTRAN Object Time Errors	13-21
	FORTTRAN-Callable Utilities	13-22
	Sample FORTRAN Deck Setups	13-24
	FORTTRAN Notes	13-27
	FTN4 Control Statement Parameters	13-30
	FTN4 PROGRAM Statement	13-34
	FORTTRAN Version 4 Notes	13-35
	Time-sharing FORTRAN	13-36
	Minnesota FORTRAN (MNF)	13-37
	Rational FORTRAN (RATFOR)	13-39
14	COBOL	14-1
	COBOL5	14-2
	COBOL5 Parameters	14-3
	Execution Memory and Time	14-4
	COBOL 5 versus COBOL 4	14-5
	COBOL 4 to COBOL 5 Conversion Aid	14-8
	COBOL Compilation Errors	14-10
	COBOL Notes	14-10
	COBOL 4	14-11
	COBOL 4 Parameters	14-11
	Conversion to COBOL From IBM	14-13
	COBOL 4 Notes	14-17
	Sample COBOL Deck Setups	14-20
	Overlay for COBOL 4	14-22
	Compile with Subroutines and Overlay	14-23

15	ERRORS, DUMPS AND DEBUGGING	15-1
	Error Messages	15-2
	Loader and NOS/BE Errors	15-2
	Tape Parity Errors	15-2
	MSS Errors	15-3
	Mode Errors	15-4
	Reading Dumps - FORTRAN Debugging	15-5
	Job Rerun	15-6
	Abort Processing	15-7
	CP Time Limit	15-7
	IO Time Limit	15-7
	Mass Storage Limit	15-7
	Message Limit Exceeded	15-7
	Reprieve	15-7
	File Dumps	15-8
16	SOURCE LIBRARY MAINTENANCE (UPDATE)	16-1
	UPDATE	16-1
	UPDATE Control Statement Parameters	16-1
	UPDATE Directives	16-3
	Sample UPDATE Setups	16-4
17	OBJECT LIBRARY MAINTENANCE (EDITLIB)	17-1
	EDITLIB	17-1
	EDITLIB Command	17-1
	EDITLIB Directives	17-2
	EDITLIB Examples	17-5
	FORTRAN Overlay in a Library	17-9
18	SUBPROGRAM LIBRARIES	18-1
	NOS/BE System Libraries	18-1
	Libraries of Subprograms	18-3
	SYSMISC	18-3
	ARLNALG	18-4
	CONMIN	18-4
	EISPACK	18-4
	FUNPACK	18-5
	IMSL	18-5
	LINPACK	18-6
	MINPACK	18-6
	MSL	18-6
	NSRDC	18-7
	NSRDC5	18-8
	SANDIA	18-8
	SHORTIO	18-9
	SSP	18-9

Plotting Libraries	18-10
Printer Plots	18-10
Off-line Plotters	18-10
Calcomp Plotters	18-11
Calcomp 936 Pen Plotter	18-12
Calcomp Functional Package	18-13
SC4020 to Calcomp	18-15
Calcomp Three-D Software	18-16
DISSPLA	18-17
DISSPOP: The Post-Processor Option for DISSPLA	18-19
Examples	18-20
 19 PROGRAM LIBRARIES AND OTHER ROUTINES	 19-1
Libraries of Main Programs	19-1
BIMED	19-1
MNSRDC	19-1
UTILITY	19-2
Miscellaneous Programs	19-3
Other Programming Packages	19-4
BIMEDP	19-5
Other Sources of Programs	19-6
Program Documentation	19-8
Source Code	19-8
Other Versions of System Software	19-9
 20 SPECIAL-PURPOSE PROGRAMMING LANGUAGES	 20-1
ABAQUS	20-1
ALGOL5	20-2
APL	20-2
Automatically Programmed Tools (APT)	20-3
BASIC 3	20-4
COMPASS	20-4
Checkpoint/Restart	20-5
Operator Checkpoint	20-7
Data Management System 170 (DMS170)	20-8
GPSS	20-9
MIMIC	20-9
NASTRAN	20-10
OMNITAB	20-12
Pascal 6000	20-12
PL/I	20-12
SHARP	20-13
Simscrip II.5	20-14
Snobol	20-14
Sort/Merge 4	20-15
Sample Sort/Merge 4 Setups	20-16
Sort/Merge 5	20-19
Sample Sort/Merge 5 Setups	20-22
SPSS	20-23

System 2000, Version 3.0	20-25
System 2000, Version 2.60	20-30
Text Processors	20-32
RNF	20-32
PROSE	20-32
 21 USER HELPS	 21-1
Conversion Aids	21-1
Communications with Operator	21-2
PM - Carriage Control Disposition at	
Remote Sites	21-2
Dayfile Message Display	21-4
Reserved Words	21-4
Estimating Print Requirements	21-5
Generating End-of-Record Marks	21-5
 22 INTERACTIVE GRAPHICS	 22-1
Tektronix 4015/4014	22-2
Tektronix 4662 Plotter/Digitizer	22-4
Tektronix 4051	22-6
Tektronix 4027 Color Terminal	22-7
EZGR - Color Graphics	22-8
Interactive Data Display System (IDDS)	22-9
MOVIE.BYU	22-10
NASTEK	22-10
 23 OTHER COMPUTER CENTER EQUIPMENT	 23-1
Datagraphix Mini Autocom Microfiche System	23-1
Calcomp model 936	23-3
Calcomp Plot Request	23-3
Xerox-8700 Printer	23-7
 24 INTERFACING WITH OTHER COMPUTERS	 24-1
Navy Laboratory Computer Network (NALCON)	24-1
NEMS	24-2
PC Interface	24-3
XMODEM Micro-to-Mainframe File Transfer	24-3

A	Appendix A	A-1
	DTNSRDC Character Set	A-1
	ASCII Character Set	A-3
B	Appendix B	B-1
	Keypunch Characters	B-1
	Use of Alternate Keypunch Characters	B-1
C	Appendix C	C-1
	Printer Carriage Control	C-1
	Perforation Skipping	C-2
	Print Density Control	C-2
	User-programmable 580 Printer Control	C-3
D	Appendix D	D-1
	Internal Data Structure	D-1
E	Appendix E	E-1
	Octal-Decimal Conversion	E-1
	CDC CYBER Real Floating	E-1
	Octal-Decimal Fractions	E-2
F	Appendix F	F-1
	Control Statement Field Lengths	F-1
	Recommended Field Lengths	F-2
G	Appendix G	G-1
	Glossary	G-1
H	Appendix H	H-1
	Computer Reference Centers	H-1
I	Appendix I	I-1
	Computer Reference Manuals	I-1
	Index	Index-1

***** INTRODUCTION TO CDC COMPUTERS *****

The Computation, Mathematics and Logistics Department has two large Control Data Corporation (CDC) digital computers and a Mass Storage System.

The large computers include a CDC CYBER 176 and a CDC CYBER 170 model 750. The Mass Storage System (MSS), which is driven by a CDC CYBER 170 model 825, is a secondary disk archival system and is connected to the CDC computers. *

The CYBER 176 is in Building 191; the CYBER 750 is in Building 17; the MSS is in Building 193. The Dispatch Office (for submitting card decks and receiving output at Central Site) is in Building 17.

The term 'CDC CYBER' refers to any of the CDC computers, unless otherwise qualified.

Normally, classified jobs are run during classified time on the CYBER 750.

Each large CDC computer has a single Central processing unit (CPU) and 1000000 octal (262,144 decimal) 60-bit words of memory, but only 400000 octal is user-addressable.

Each CPU has 24 registers for information on which to operate: 8 address (A) registers, 8 operand (X) registers and 8 increment (B) registers. The CYBER 176 and CYBER 750 CPU's can perform several different functions simultaneously (2 adds, 2 multiplies, 2 increments, 1 divide, 1 shift, 1 boolean, 1 branch) and each has a buffer of 12 CM words of instructions, called an instruction stack. The CYBER 176 and CYBER 750 stacks clear differently.

Peripheral processors (PP's) are small computers (memory of 4096 12-bit words) which handle most input and output (I/O). There are 20 PP's each on the CDC computers. The CYBER 176 also has 4 FLPP's (first level PP's).

* - The MSS is described in a separate publication: "Computer Center Mass Storage System User's Guide", CLMD-82-19.

Each CDC computer has 24 I/O channels. All peripheral equipment, including printers, tape and disk drives, and multiplexors for remote terminals, interfaces with the Central system through the PP's via the I/O channels. The CYBER 750 and the CYBER 176 have two channels each for tape drives.

The CYBER 176 and 750 are compatible; binary programs generated on one system may be run on the other. However, each has its own permanent file system. Hence, users often choose one system for processing a class of programs to avoid maintaining and paying for duplicate sets of permanent files. An alternative is to use the MSS and access the files from any computer.

*** Hardware Configuration ***

CDC CYBER 170 model 750 (mainframe E - MFE)

- 262,144 60-bit word memory
- 20 peripheral processors
 - 2 model 885 (fixed) disk drives
 - 2 model 844-41 disk drives
 - 4 model 844-21 disk drives
 - 2 model 677-3 seven-track tape drives
 - 4 model 679-5 nine-track tape drives (1600/6250 bpi)
 - 2 model 679-3 nine-track tape drive (800/1600 bpi)
 - 1 model 405 card reader
 - 1 model 415 card punch
 - 2 model 512 line printers (1200 lpm)
 - 1 model 580-20 line printer (2000 lpm)
 - 1 2550 data concentrator with the following lines:
 - 4 hard-wired 9600-baud terminals
 - 1 hard-wired 4800-baud terminal
 - 7 telephone lines for ASCII/BCD 4800-baud terminals
(202) 227-3157
 - 1 telephone line for ASCII/BCD 2400-baud terminals
(202) 227-3124
 - 1 telephone line for ASCII/BCD 2000-baud terminals
(202) 227-3190
- 40 telephone lines for 1200-/300-baud interactive terminals
 - Bell -- (202) 227-3000 (30)
 - Vadic -- (202) 227-3500 (10)
- 2 telephone lines for 1200-baud interactive terminals
 - Annapolis -- (301) 267-2025
- 6 telephone lines for 300-baud interactive terminals
 - Annapolis -- (301) 267-2011

CDC CYBER 176 (mainframe F - MFF)

262,144 60-bit word memory
524,288 60-bit words of LCME
4 first level peripheral processors (FLPP)
20 peripheral processors
4 model 885 disk drives (permanent files)
4 model 844-41 disk drives (system use)
4 model 844-21 disk drives (user device sets)
2 model 819-21 disk drives (swapping)
2 model 677-3 seven-track tape drives
2 model 679-3 nine-track tape drives (800/1600 bpi)
4 model 679-5 nine-track tape drives (1600/6250 bpi)
1 model 405 card reader
1 model 415 card punch
1 model 512 line printer (1200 lpm)
1 model 580-20 line printer (2000 lpm)
1 VAX 750 mini computer for access via NALCON
1 6774 data concentrator with the following line:
1 hard-wired 40.8 KB terminal
2 2551-2 data concentrators with the following lines:
2 hard-wired 9600-baud terminals
3 hard-wired 1200-/300-baud interactive terminals
16 telephone lines for ASCII/BCD 4800-baud terminals
(202) 227-4740

56 telephone lines for 1200-/300-baud interactive terminals
VADIC (202) 227-4800 (32)
VADIC (202) 227-4850 (24)

1 telephone line for 1200-baud interactive terminals
Annapolis -- (202) 267-2017

7 telephone lines for 300-baud interactive terminals
Annapolis -- (202) 267-2018

CDC 7880 storage system with

CYBER 170 model 825 (mainframe G - MFG) (Mass Storage System)

262,144 60-bit word memory
20 peripheral processors
4 model 885 (fixed) disk drives
2 model 844-41 disk drives
3 model 679-5 nine-track tape drive (1600/6250 bpi)
1 model 512 line printer (1200 lpm)
1 model 405 card reader
6 model 7881 cartridge storage units
12 model 7882 mass storage transports

*** NOS/BE 1.5 ***

The operating system for all CDC CYBER computers at DTNSRDC is called the Network Operating System/Batch Environment, version 1.5, (NOS/BE 1.5 - level 552) and differs only slightly from the standard NOS/BE system of CDC. The interactive system for teletype-compatible terminals and medium-speed remote batch terminals is called INTERCOM. The unit record equipment (card readers, line printers, etc.) at the Central Site is controlled by a subsystem called JANUS, which occupies one control point on each computer. The remaining control points are available for interactive, user batch and other system jobs.

Permanent files (user program and data files retained for frequent use) reside on model 885 disk drives. Most of the operating system resides on model 844 system disks. User files, if not specifically requested on a tape or user device set, will be assigned to available disk areas.

***** USER INTERFACE WITH THE COMPUTER CENTER *****

*** General Information ***

Consultation is available from the User Services Group:

Carderock: Bldg 17, Room 100, (202) 227-1907

Annapolis: Bldg 100, Room 1T, (301) 267-3343

For information about:

services available - Code 1892.1 (202) 227-1907

accounting - Code 189.3 (202) 227-1361

hardware - Code 1894 (202) 227-1400

The ADP Control Centers are located at the Central Site of each CDC computer. You may submit decks and pick up output as well as obtain information on the progress of your jobs from the ADP Control Center.

Computer Center Notes is a publication sent to all registered users. It is sent whenever there is information to be disseminated. The System Bulletin provides current news and is printed at the beginning of each batch job and each Intercom LOGIN (except SUP).

*** Trouble Forms ***

A Trouble Form is used:

- 1) for refund requests (page 2-2; POLICY)
- 2) when problems are encountered
- 3) for suggestions, gripes and complaints.

The Trouble Form should include a succinct description of the problem and include as much documentation (dayfile, listings, dumps) as possible. It should be submitted to Code 1892 for processing.

Trouble Forms not requiring documentation may be entered directly into the computer from either batch or Intercom. The procedure GRIPE (page 6-5; CLIB/P) is used for this.

*** Gripes ***

Gripes and comments may be sent to User Services via Intercom. The GRIPE procedure (page 6-5) may be used (batch or Intercom). All messages and gripes will be read and Trouble Forms prepared, if appropriate.

*** Refunds ***

Refunds on lost time are defined in POLICY. Requests must be accompanied by output of the run and a Trouble Form, and should be filed within five working days. Decisions on refunds will be made by Code 189.

*** ADP Control Center ***

The ADP Control Center has the following capabilities:

- 1) Reproduce binary, BCD (026) or EBCDIC (029) decks. The decks cannot contain embedded end-of-information (6/7/8/9) cards.
- 2) List decks. The decks cannot contain embedded end-of-information cards.
- 3) Convert 029 decks to 026. The Control Center must be informed if the input is EBCDIC. The decks cannot contain embedded end-of-information cards.

To convert 026 decks to 029 decks at Central Site, use 'EC=029' on the ROUTE command (page 5-12). To convert to 029 at a remote punch (1700), a special procedure, CV029 (CLIB/P), is available. The decks to be converted cannot contain embedded end-of-record or end-of-information cards.

To reproduce a deck with sequencing, gangpunching or alteration of fields, see CLIB/U: CARDS/CARDS2.

The following EAM facilities are available off-line:

- 1) Card interpreter (available at Annapolis)
A small card interpreter is available at Central Site
- 2) Shredder (available at Central Site)
- 3) Card Sorter (available at Annapolis)

Telephone numbers for the ADP Control Center are:

Carderock - (202) 227-1435
Annapolis - (301) 267-3340

*** Central Site EAM Processing ***

At the Central Site, card listing and reproduction are performed without additional charge except as indicated below.

- 1) EAM jobs of less than 2000 cards may be processed during the prime shift.
- 2) EAM jobs of greater than 1999 cards will be processed overnight. Jobs of this size which must be processed during the prime shift will be charged to the user's job order number.
- 3) The Computer Center reserves the right to charge at the established rate for jobs listing more than 4000 cards. Reproduced decks of over 500 cards will be charged to the user's job order number.
- 4) Systems utilities which service EAM functions may, of course, be utilized by users. See User Services Group.

*** Punched Cards from Central Site ***

All end-of-record and end-of-file cards punched should have been marked by ADP Control with a red marker on the end. In addition, any card which triggered a compare error will be marked. The user should pull out any red marked defective cards, since the card was repunched correctly by the system immediately following the bad one.

In addition to what you punched, the deck will also contain a leading eye-readable banner card and EOR/EOI cards at the end. These are normally removed before using the deck.

***** BATCH PROCESSING *****

This chapter begins with a general discussion of batch job processing (3-2). It continues by describing remote batch job processing (3-3 ff). It ends with a discussion of a special time allotted to individual users by the wall clock hour known as Block Time (3-8).

*** Job Processing ***

A job entering the system goes to the input queue and waits for the system to check that the resources required by the job (e.g., central memory (CM) and tape drives) are available. When a job has the highest priority (priority increases as a job waits) and resources are available, the job is assigned a job descriptor table (JDT) entry and may occupy one of the control points for active jobs. The job is loaded into an assigned area of CM called the field length (FL). The job may not access any area of memory outside its own FL, but the system will vary the size of FL during execution. A user may vary the the maximum FL by the RFL control statement (page 8-9). When a job is waiting for resources such as FL, and central processor (CP) time, it may be swapped out to allow other jobs to run. The job will share the CPU and the PP's with the other active jobs until it is completed. Then the results of the job wait in the output queue for printing or punching.

You control your job's execution through control statements which must appear in the first logical record of your job. One statement is processed at a time and any control statements requiring data (such as a FORTRAN compilation) will take the next logical record in the job's input stream.

A record of the activity on the system is recorded in the dayfile. The last page of output for every job contains that job's dayfile, listing all the control statements processed (except the CHARGE card) and the system messages.

The CYBER 176 and 750 systems operate independently, but the operating systems are compatible. Each computer has its own permanent file base. If you must have the same information on both computers, you should create duplicates or keep the files on the Mass Storage System, so that jobs are not aborted due to the unavailability of such files.

Policy governing the use of the computers and the charges for different categories of service is given in the "Computer Center Policy". Information on current charges may be obtained from Code 189.3, (202) 227-1361.

The CYBER 750 is used during classified time to process jobs for classified projects.

***** Remote Batch Processing *****

In addition to Central Site, there are many medium-speed (usually dial-up) remote batch entry (RJE) terminals. These terminals include at least a card reader (for submitting batch jobs) and a line printer (for printing job output).

*** Medium-speed Terminals ***

Medium-speed terminals (ASCII or BCD) operate under the control of Intercom and consist of a card reader, a line printer and usually a CRT display. The CDC 200 User Terminal (200-UT) can be connected directly to the CYBER computers. A number of other terminals (including Harris 1600 *, CDC CYBER 18, DEC VAX, CDC 734 and Data 100) have software packages (both ASCII and BCD) which emulate the 200-UT. All are dial-up at 4800 baud (see page 1-3, 1-4 for the telephone numbers).

A set of remote batch processing commands (page 3-4) allows for reading and printing files. Print files may be ROUTED to a medium-speed terminal from batch jobs or interactive terminals. Operators of 200-UT-type terminals may alter priority, divert or drop jobs in their queues.

On medium-speed terminals, the entire deck must be punched in only one mode since a switch or type-in tells the terminal to read 026 or 029 mode cards. No binary cards can be read.

Since medium-speed terminals generally do not have a card punch, users who wish to punch cards must ROUTE them to Central Site.

At 4800 baud, a terminal can read about 300 cards per minute and print about 300 lines per minute. Therefore, it is recommended that card decks be limited to one box and printing to 50 pages.

When printing, the only carriage control characters effective at a 200-UT are:

blank	single space before printing
1	eject to top-of-page before printing
0	double space before printing
-	triple space before printing
+	suppress spacing before printing (print on same line)
PM	printer message (to stop the terminal to allow a new printer setup)

All others are treated as blanks.

Most medium-speed terminals may be used for local card listing.

* - See "Harris Operator's Guide", CMLD-82-15.

*** Medium-Speed Batch Processing Commands ***

The following commands are available at a medium speed 200-UT-compatible terminal for processing batch jobs and output files.

** Printer **

ON,LP1 Turn the line printer on. Must be the first command after
ON LOGIN.

OFF,LP1 Turn the printer logically off. ON,LP1 is required before
OFF printing may resume.

END,LP1 End printing. Discard the remaining output. GO,LP1 to
E,LP1 print the job's dayfile. Ignore repeat count, if any. A
END second END,LP1 will stop printing the dayfile.
E

GO,LP1 Resume printing after wait or some error conditions.
G,LP1
GO
G

CONTIN Resume printing when transmission was stopped because a
C message was received (and is displayed on the CRT) or the
interrupt key INTER was pressed.

BSP,LP1,n Backspace line printer <n> (times 10 octal) sectors.
BSP,,n (default: 1 (10 octal sectors))
BSP

REP,LP1,m Reprint <m> (octal) additional copies (default: 1; max: 37
REP,,m octal). The repeat count (RC) is cumulative (i.e., REP,,2
REP followed by REP,,3 is the equivalent of REP,,5). <m> may
not be negative.

REW,LP1 Rewind the print file and turn printer logically off.
REW ON,LP1 is required before printing may resume. The
residual repeat count is saved.

RTN,LP1 Rewind the print file and return it to the output queue.
RTN The residual repeat count is saved.

WAIT,LP1 Temporarily halt printing. To resume, use GO,LP1.
WAIT

DEFINE,LP1,FCxx
DEFINE,,FCxx Establish forms code for printer (<xx> is the forms code,
e.g., 1T (see page 5-14)). The printer must be prepared
before this command is issued.

DEFINE,LP1 Restore default printer settings. The printer should be
DEFINE restored before this command is issued.

** Card Reader **

ON,CR1 Turn card reader on.

OFF,CR1 Turn card reader logically off. ON,CR1 is required before reading may resume.

END,CR1
E,CR1 End card reading.

GO,CR1
G,CR1 Resume card reading after wait or some error conditions.

CONTIN
C Resume reading when transmission was stopped because a message was received (and is displayed on the CRT) or the interrupt key INTER was pressed.

READ Read cards into the input file for this terminal.
 (abbreviation R not allowed starting at level 508)

WAIT,CR1 Temporarily halt card reading. To resume, use GO,CR1.

** Printer and Card Reader **

WAIT,ALL Temporarily halt reading and printing. To resume, use GO.

GO,ALL
G,ALL Resume reading and/or printing which was halted by WAIT.
 Also required, with or without equipment, when any of the following errors have occurred:
 DEVICE NOT READY
 JOB STATEMENT ERROR
 INPUT FILE ERROR
 OUTPUT FILE ERROR

CONTIN
C Resume reading and/or printing when transmission was stopped because a message was received (and is displayed on the CRT) or the interrupt key INTER was pressed.

** Queue and Status Displays **

H,I Display input queue for this terminal.

H,O Display output queue for this terminal.

H,E Display jobs in execution for this terminal.

H,S Display current status of of this terminal's I/O devices.

H,P Display punch queue for this terminal. (Jobs in the punch queue must be diverted to high-speed terminal or central site for punching. See below.)

**** Manipulating Queue-resident Files ****

In the following commands, jobname is the name of the job/file affected. It may be the 7-character job name or the last 2 characters of the NOS/BE job name.

DIVERT,jobname,tid,q

Transfer files/jobs to another location.

jobname - name of job to be transferred.

tid - ID of terminal to receive diverted job. If omitted, divert to Central Site.

q - divert jobs from this queue only. <q> may be 0 (print), P (punch), I (input), or omitted (all queues).

Examples: DIVERT,XXXXOR7,555,0 Divert print file
XXXXOR7 to terminal 555.
DIVERT,R7,,P Divert punching of job R7
to Central Site.

DROP,jobname

Drop a job while it is in execution. Any output is placed into the output queue.

EVICT,jobname,q

Eliminate job from specified queue. <q> may be I (input), 0 (print), P (punch), or omitted (all queues).

PRIOR,jobname,p,q

Change priority of a print or punch file. <p> is a 1- to 4- digit positive octal number defining the new priority. Jobs with priority 0000 will remain in the queue. <q> may be 0 (print), P (punch) or omitted (print and punch).

** Some Error Messages **

COMMAND IGNORED

The command cannot be processed. (E.g., OFF,LP1 cannot be executed when the printer is already off.)

COMMAND REJECT

Either a GO or ON command must be entered before other commands may be processed.

CRn,jjjjjjj,INPUT FILE ERROR

Errors in creating the input file. Job <jjjjjjj> is dropped from the system. Check cards for errors. END,CR and try again.

CRn,jjjjjjj,SHIFTED DATA,INPUT FILE ERROR

A card was read incorrectly. To proceed: END,CR1; check for a damaged card in the last group read and fix; put an EOI card (6/7/8/9) in front of the job card (for safety) and GO,CR1 to read complete deck.

CRn,jjjjjjjjjj,JOB CARD ERROR

The job card is incorrect. <jjjjjjjjjj> is the job name or the first 10 characters of the job card. Return to the user for correction.

CRn,NOT READY

Place more cards (or an end-of-file card) in the reader. Press LOAD. Enter GO,CRn.

INPUT SUSPENDED BY SYSTEM

System input queue is overloaded. Reading resumes automatically when the overload condition ends. (The printer must be logically on.)

LPn,NOT READY

Correct the condition. Ready the printer. Enter GO,LPn.

*** Block Time ***

Block time is a special period of time, usually on the third shift and on weekends, when computer use is reserved for a specific job or group of jobs. The user is charged by the wall clock instead of by the usual charging algorithm.

To schedule and use block time:

1. Call the Operations Branch ((202) 227-1365) and be prepared to supply the following information:
 - jobname (the first 5 characters of the jobname of job(s) to be run) and number of individual jobs to be run
 - approximate wall clock time
 - resources required
 - if more than one job, can they run simultaneously?
 - may the jobs be run on either system (all needed files are on the MSS)?
 - may the job be checkpointed by the operator?
 - tape drives
 - disk drives for user device sets
2. After time has been scheduled, put the job(s) into the input queue at priority zero ('P0' on the job card).

The minimum block time which may be scheduled is one hour for the third shift and four hours for a weekend day. Time is allocated on a first-come, first-served basis. Weekend block time should be scheduled by Thursday noon so that overtime can be scheduled for the operators. Weekend completion of block time scheduled after Thursday noon is not guaranteed.

If jobs exceed their estimated time plus one hour, the Computer Center reserves the right to checkpoint or drop them. If your job is checkpointed, you must, within 24 hours, schedule time and submit a RESTART deck (page 20-5) to complete the job. This is necessary because one or more tapes have been provided for the checkpoint and you may not keep them.

For further details, see POLICY.

Suggestion

When a job is checkpointed, everything related to the job (memory and scratch files) is written onto the checkpoint tape(s). To minimize checkpoint size, return files when they are no longer needed.

***** INTERCOM 4 *****

*** Introduction ***

Intercom permits simultaneous remote access to the central computer resources in an interactive mode for users at teletype-compatible or medium-speed remote batch terminals. Intercom operates concurrently with the batch mode activity at the central computer site and gives terminal users the ability to

- . Create programs by entering source statements at the terminal.
- . Create, store, reference and edit disk files.
- . Submit FORTRAN Extended, COBOL, System 2000 natural language, Algol (the same compilers as used for regular batch jobs), or Basic programs for compilation and execution under Intercom control.
- . Converse interactively with executing programs from the terminal.
- . Submit jobs to the NOS/BE batch queue for processing.
- . Enter NOS/BE control statements (utilities) for processing.

Intercom facilities are described in detail in the Intercom Reference Manuals (see page i).

*** Sign-on Procedures ***

Each Intercom user must obtain an access number from Code 189.3 for his job order number before having access to the system.

Most interactive terminals are connected to the Central Site by data sets. Many ports are available. If the computer is down during prime shift, the terminal may automatically transfer to the recorded status message.

The following telephone numbers are of interest:

- (202) 227-3043 recorded computer status message
- (202) 227-1907 User Services (Carderock) for assistance
- (301) 267-3343 User Services (Annapolis) for assistance

See pages 1-3, 1-4 for the computer telephone numbers.

Detailed procedures for connecting with the computer vary with the type of remote terminal and the data set and are not given here. In general, the following steps are necessary:

1. Turn on the electrical power to the terminal.
2. Set on half-duplex.
3. Call the computer (phone numbers on pages 1-3, 1-4).
4. When the high, steady data tone is heard, complete the phone connection by either:
 - pushing the data button on the data set -or-
 - placing the receiver in the acoustic coupler -or-
 - other procedure for certain terminals.
5. Press RETURN (or SEND) key to inform the system of the baud rate (1200 or 300). The computer will show its identification, the date and time.
6. Enter LOGIN. (the period is optional)
7. In response to "ENTER LOGIN NAME-", enter xxxxyyyyyy where xxxx is your registered user initials, yyyyyy is usually your last name (or first six letters of it).
 - 7a. One of the short forms for 6 and 7 may be used:
LOGIN, xxxxyyyyyy
LOGIN, xxxxyyyyyy, SUP
8. In response to "XXXXXXXXXX ENTER ACCESS NUMBER", enter the access number corresponding to the job order number to which the session is to be charged. (Only previously registered user name/access number combinations will be accepted. If you must work under several job order numbers, you must obtain an access number and Intercom ID for each.)
9. In response to "XXXXXXXXXX ENTER TURNKEY PASSWORD", enter it in the blackened out area. If your turnkey is the default, or if it hasn't been changed in 90 days, you will be prompted to change it.
10. When all entries are valid, the system will display the date, time, terminal ID, equipment and port numbers, and, perhaps, a brief system bulletin.
11. The system prints "COMMAND- " when it is ready to begin work.

*** User Turnkey Password ***

An additional level of security is available through the use of the Intercom TURNKEY command which will define a 6- to 9-character User Turnkey Password which must be specified at future LOGINs. The turnkey offers protection against unauthorized charges to the user's Intercom account. It must be changed from time to time (at least every 90 days). The Intercom turnkey does not apply to permanent files or batch usage.

Registration for Intercom gives you access to all unclassified CDC systems and establishes your default turnkey password. You should define your own turnkey password on both the CYBER 176 and the CYBER 750.

The TURNKEY command has no parameters. You will be prompted for passwords as follows:

```
XXXXXXXXXX enter old turnkey password
XXXXXXXXXX enter new turnkey password
XXXXXXXXXX enter turnkey again for verification
```

Enter the requested turnkeys in the blackened space. The new password may not be the current one, the one before, or the default (the one you received when you registered for Intercom).

You must remember your turnkey, since the Computer Center has no record of it. If forgotten, the Intercom ID must be re-established by User Services to reset the turnkey.

When the Intercom password file is rebuilt (e.g., at the start of each fiscal year), this turnkey may need to be re-defined.

*** SECURE/RESUME a Terminal ***

The Intercom SECURE command will secure a terminal if you must leave for a few minutes and don't want to LOGOUT or disconnect. This will prohibit others from using the session while you are away.

To resume the session, enter RESUME, which will request the username, access number and, the user turnkey password. This may be shortened with 'RESUME,<username>'. If the session is not resumed within 15 minutes, the terminal is disconnected for inactivity, as usual. Three unsuccessful attempts to RESUME are considered a security violation. On most terminals, bells will ring! Automatic LOGOUT occurs 30 seconds later.

*** Sign-off Procedures ***

To terminate your session or when requested by central operator, type 'LOGOUT.' (the period is optional). If you are in EDITOR, you must type BYE before LOGOUT. A time summary of the session and estimated cost will be printed. When the summary has printed, finish the sign-off by turning off the teletype and hanging up the data set (push the talk button) unless another user is waiting to LOGIN.

*** System Bulletin ***

A System Bulletin may be typed at LOGIN time. If 'LOGIN,...,SUP' was used, only critical bulletins will be displayed. SYSBULL may be executed at any point during an Intercom session. Except for 'SYSBULL, LOGIN', CONNECT,OUTPUT is required to print at the terminal. (See page 5-9: SYSBULL)

*** Correcting and Interrupting ***

1. To delete a character, hit the <backspace> key (or press the <ctrl> key and type the letter H simultaneously), then retype the character. On some terminals, the carriage will not move, nor will a character be typed to indicate your correction. You must keep track of the changes.
2. To cancel the entire current line, press the <ctrl> key and type the letter X simultaneously. Re-type the line correctly or press carriage return and the type the line again. <ctrl>-X does not generate another prompt.
3. To exit from the current command:
 - if the system is typing, hit ESC key (or interrupt key), then %A, then return
 - if the system is not typing, enter %A, then return.After user abort, the system will reply COMMAND- or .. for the next command.
4. Hitting the ESC key, followed by %S and return, will cause printing to jump to the next buffer, but will not abort execution of the current command.
5. To delete a line from the edit file, the delete command must be used.
6. If you are accidentally disconnected from your terminal (e.g., hit EOT (<ctrl>-D) by mistake), you will be able to LOGIN within 15 minutes with all files preserved. If under NETED or editor, the text buffer should be intact.
7. If execution of a command or directive exceeds 60 CP seconds (MFE), or 20 CP seconds (MFF), that command is aborted. (see 4-19: ETL)
8. When a program is accepting input from the terminal, you can simulate end-of-record or end-of-file by entering %EOR or %EOF, respectively. To simulate end-of-information, enter %EOF twice. This allows you to enter data for programs (both user programs and system programs such as EDITLIB, etc.) directly at the terminal. See page 21-5 for the different representations of end-of-record and end-of-file.
9. To enter any command from keyboard while in TAPE,ON mode (page 4-29), the user must end the line by carriage return, line feed, X-OFF (hold <ctrl> key while typing letter S).
10. When any EDITOR command gives a loader error, check that it was not incorrectly typed with a period (.) at the end.

*** Intercom User Limitations ***

1. Registration of a job order number is a prerequisite for Intercom usage. In addition, each individual must receive Intercom training and password assignment.
2. The maximum CPU time per command or program execution is 60 seconds (MFE) or 20 seconds (MFF) *. The maximum memory to load a user program for execution is 200000 octal words. Remote batch terminals are limited to 65000 octal words. Permission to alter field length down from 200000 or time limit up (to 500 octal (320 decimal) seconds) or down is granted (see page 4-19: EFL, ETL).
3. When a file used during a session is no longer needed, type in 'RETURN,lfn' to delete it from user local files. It is especially important to return permanent files as soon as possible since there is a total system maximum of 200 attached permanent files at any one time. Exceeding this will hang the system for all users! See also pages 5-8: CLEAR and 5-9: RETAIN.
4. Use of the ASSETS or SUMMARY commands while in EDITOR will not include the CP time of the current EDITOR session in the totals.
5. Attempt to LOGIN when Intercom is not available or the system is hung results in a data tone that drops out or the type head chatters with no typing or the phone rings without answering. For a recorded message about the system status, call (202) 227-3043.
6. Occasionally, the system will go down without notifying users or before users can catalog current local files. If this happens, try to LOGIN again within 15 minutes and use the FILES command to see if your local files have been retained from the previous session. If so, work may be continued and files cataloged.
7. Binary programs being executed through Intercom must not be attached with local file names identical to any medium-speed remote batch terminal commands (see pages 3-4 ff). Avoid single letter names, the commands these abbreviate, standard system control statements (such as FTN5, COPY, etc.) and certain other DTNSRDC program names (LX, MM, PK, SF).
8. All files, including the special files INPUT, OUTPUT, PUNCH and PUNCHB, are returned automatically at LOGOUT.
9. The only command which automatically connects INPUT or OUTPUT is the EDITOR 'RUN' command. 'RUN,<compiler>' connects both INPUT and OUTPUT. 'RUN,<compiler>,NOGO' connects OUTPUT.
10. The use of DMP will disconnect OUTPUT.

* - On MFE, time is in CYBER 750 seconds; on MFF, it is in CYBER 176 seconds.

*** Program Construction - NETED ***

The NETED editor * may be used to create and modify programs and other sequential files at the terminal. All new files saved under NETED are on permanent file devices. NETED can edit lines of up to 140 characters. Lines in NETED are not numbered; position within the file is controlled by moving a pointer forward and backward in the file. Any source language which specifies certain columns (e.g., FORTRAN statements in column 7) must either be spaced or tabbed over to the correct column. The default tab character is a semi-colon (;). The tab character and stops are set by the STAB command.

When in NETED, NOS/BE commands cannot be executed. To compile a program, it must first be saved. The compiler must then be called explicitly. You are responsible for rewinding the input file and 'LGO'. Procedures RUNFTN5, RUNFTN, RUNCOB5, and RUNCOB may be used to simulate the EDITOR (4-9) 'RUN,FTN5', 'RUN,FTN' and 'RUN,COB' commands.

Note that when %A is executed during NETED, the program is terminated and the workfile is lost.

A document describing NETED may be obtained by:

BEGIN,DOCGET,,OTHER,,NETED,OUTPUT,MSACCES=<pw>.

Note: Only the lines to be typed by the user appear below.

** FORTRAN 5 **

```
ATTACH,NETED,NETEDF      <-- FORTRAN tabs, no prompting
NETED,EX                 <-- enter NETED in input mode
;PROGRAM EXAMP (INPUT=128, OUTPUT=128, TAPE5=INPUT)
C  author and address
C  uses list-directed I/O  for comma-separated, unformatted data
;CALL CONNEC (5)
;CALL CONNEC (L"OUTPUT")
;PRINT *, 'Type in  a,b,k '
2;READ(5,*) A, B, K
;IF (A .EQ. 0.) GO TO 6
;C = A ** K + B
;PRINT 4, C, A, B, K
4;FORMAT (1X, 3F7.2, I3)
;GO TO 2
6;STOP
;END
.                         <-- switch to edit mode
PA                        <-- list for proofreading
SAVE                     <-- make user local file
BEGIN,RUNFTN5,,EX       <-- compile and execute
```

* - NETED was obtained from the Lawrence Berkeley Laboratory and was enhanced by one of our former customers (CERL). NETED is now maintained by User Services.

** Card Image FORTRAN Deck **

```

ATTACH,CARDS,ID=xxxx      <-- bring permanent file to local files
ATTACH,NETED,NETEDF
NETED,CARDS               <-- enter NETED in edit mode
I      PROGRAM TEST (INPUT=128,OUT=128)
C/SIN/COS/ $             <-- text replacement, SIN into COS calls
                           (will also change any 'SIN' to 'COS')
T      <-- go to top of file
L PROG                   <-- look for PROGRAM statement
C/OUT/OUTPUT/            <-- correct file name in PROGRAM statement
L <unique>               <-- position pointer at next line with the
                           <unique> test string
D 2                      <-- delete lines
PA                      <-- list for proofreading
SAVE NEWPRO              <-- make user local file and leave NETED
BEGIN,RUNFTN5,,NEWPRO    <-- compile and execute revised program
STORE,NEWPRO,xxxx        <-- catalog corrected program

```

** Execution - External Files Required **

```

...                      <-- creation or editing of user program
BEGIN,RUNFTN5,,<1fn>,NOGO <-- compile without load or execute
ATTACH,NSRDC5.           <-- get needed library (period optional)
XEQ,LDSET=LIB=NSRDC5,LOAD=LGO

```

** Binary Program Runs **

To run a binary program file interactively from Intercom, be sure that the input and output files are connected to the terminal. Use either

CONNECT,INPUT,OUTPUT as a command

or

CALL CONNEC (L"INPUT") and
CALL CONNEC (L"OUTPUT") in a FORTRAN program. (see FTN5, 7-19)

To detach a file from the terminal, use

DISCONT,1fn -or- as a command
RETURN,1fn -or-
UNLOAD,1fn

or

CALL DISCON (L"1fn") in a FORTRAN program.

*** CDC Intercom EDITOR ***

The CDC editor (called 'EDITOR') may be used instead of the NETED editor. The CDC editor requires much more memory and CP time than NETED does. EDITOR lacks some features of NETED, but also has some that NETED does not. EDITOR can edit lines of up to 512 characters. Lines to be edited must have sequence numbers.

When in EDITOR, NOS/BE commands can be executed while not in create mode. EDITOR commands 'RUN,FTN5', 'RUN,FTN' and 'RUN,COB' will compile and execute the program being edited. Local files INPUT and OUTPUT are connected to the terminal by the 'RUN' command.

See CCIRM, 3-4 ff and INT, Chapter 5.

Note: only the lines to be typed by the user appear below.

** FORTRAN 5 **

```
EDITOR          <-- enter EDITOR subsystem
CREATE          <-- set automatic line number generation
;PROGRAM EXAMP (INPUT=128, OUTPUT=128, TAPE5=INPUT)
C  author and address
C  uses list-directed i/o  for comma-separated, unformatted data
;CALL CONNEC (5)
;CALL CONNEC (L"OUTPUT")
;PRINT *, 'Type in  a,b,k '
2;READ(5,*) A, B, K
;IF (A .EQ. 0.) GO TO 6
;C = A ** K + B
;PRINT 4, C, A, B, K
4;FORMAT (1X, 3F7.2, 13)
;GO TO 2
6;STOP
;END
=              <-- turn off auto line number generation
LIST,ALL       <-- list for proofreading
SAVE,FNAME     <-- make user local file
RUN,FTN5       <-- compile and execute
```

** BASIC **

```

EDITOR
DELETE,ALL          <-- to clear edit file
FORMAT,BASIC        <-- establish BASIC editing format
100 REM Compute and print
110 REM author and address
120 LET P = 3.14159266
130 PRINT " Enter x";
140 INPUT X
150 IF X<=0 THEN 200
160 LET W = SQR(P * X)
170 PRINT " Root is" W
180 GO TO 130
200 END
RUN,BASIC           <-- execution

```

** Card Image FORTRAN Deck **

```

ATTACH,CARDS,ID=xxxx    <-- bring permanent file to local files
EDITOR
EDIT,CARDS,SEQ          <-- sequence the deck for editing
L,A                     <-- list to check sequencing, then edit
90=      PROGRAM TEST (INPUT=128,OUT=128)
/SIN=/COS/,A,U          <-- text replacement, SIN into COS calls
                        (will also change any 'SIN' to 'COS')
/OUT=/OUTPUT/,90        <-- correct file name in PROGRAM statement
D,210,220                <-- delete lines
L,A                     <-- list for proofreading
S,NEWPRO                <-- make user local file
RUN,FTN5                <-- compile and execute revised program
BYE                     <-- leave EDITOR
CATALOG,NEWPRO,ID=xxxx  <-- catalog corrected program

```

*** Leaving and Re-entering EDITOR ***

To get out of EDITOR and back into command mode, enter BYE. If you have not saved your edit file, either do so and type BYE again or just type BYE again. If there is anything in your edit file, it will be retained for next entry into EDITOR. The current mode (FORTRAN, BASIC, etc.), CH, tabs and tab character are also saved, as is the number of the last line entered or listed. Then, if you enter EDITOR again, the message 'YOU HAVE AN EXISTING EDIT FILE' will be typed and your last edit file and all settings will be available. Note that it is more efficient to exit from EDITOR before doing a string of NOS/BE commands.

*** Batch Runs from Interactive Terminals ***

To run a job that requires more than 200000 octal memory locations, over 20 seconds on MFF (60 on MFE), or has much output, batch jobs may be initiated from an interactive terminal. A complete control statement record must be constructed using NETED (or EDITOR). Save the file, then issue the ROUTE or BATCH command. To return the job output to your terminal, a ROUTE statement without a TID parameter may occur in the control statement record, or the BATCH command may include INPUT,HERE.

Example: ROUTE,OUTPUT,DC=PR,TID=146,FID=*xxxx. (see page 5-12)
sends all data currently in file OUTPUT to remote terminal 146.

** Creation of Control Statement Record **

ATTACH,NETED
NETED,SGRUN

<-- enter NETED in input mode

xxxxTE,CM70000,T500. name / code <-- your job card
CHARGE,xxxx,<accessnbr>. <-- terminators required on all job
ATTACH,FNAME,ID=xxxx. <-- control statements
FTN5,I=FNAME,GO. <-- where FNAME is the file containing FTN5 source
EOR <-- will generate an end-of-record (like 7/8/9 card)
 (your input data cards)

...

PA

<-- switch to edit mode

SAVE

<-- list for proofreading

ROUTE,SGRUN,DC=IN

or

<-- save control statements for batching

BATCH,SGRUN,INPUT,HERE, For Central
Site, use TID=C in the ROUTE command or
omit 'HERE' from the BATCH command. After
completion of this command, file SGRUN is
no longer a local file.

**** Terminal Access to Batch Job Output ****

To follow the progress of a submitted job, use the 'J,xxxx' (page 4-14) or the Intercom FILES command.

When the remote output file matching the first five characters of your job name appears, the job has been executed. To bring the remote output into a local file:

BATCH,xxxxTnn,LOCAL <-- where xxxxTnn is the 7-character remote
 job name

The file is now available for NETED, EDITOR or PAGE manipulation at the terminal, or for routing to a printer.

**** Send Job to a Printer After Terminal Access ****

REWIND,xxxxTnn.
ROUTE,xxxxTnn,DC=PR,TID=C,FID=*xxxx. to Central Site -or-
ROUTE,xxxxTnn,DC=PR,TID=<tid>,FID=*xxxx. <tid> is 200UT-type remote ID
BEGIN,XEROX,,xxxxTnn,xxxx. to Xerox 8700 on 11x8.5

**** Removing Jobs in Your Queues ****

If you ROUTED a job to your input queue and now wish to get rid of it type MYQ or FILES to see where the job is. Then do one of the following:

For a job in the input or output queue:

EVICT,jobname Remove the job from your input queue.

For a job in the execute queue:

DROP,jobname Drop your executing job. The message 'OPERATOR DROP' will appear in the job's dayfile.

KILL,jobname Drop your executing job and get rid of all output. USE THIS WITH CAUTION, because you will not get any output, not even the dayfile.

Remember that these commands work only with jobs which are in your queues. If you routed the job to another terminal ID (including Central Site), the job must be removed by the other terminal or by the Central Site operator (see page 4-17).

*** Locating Jobs in the Queues ***

Three commands are available for locating batch and interactive jobs in the various queues: MYQ, FIND (or J) and Q. All lists identify the mainframe (MFE=CY750; MFF=CY176).

** MYQ Command **

MYQ Search all queues for jobs belonging to the user issuing the request, that is, those jobs with the same terminal ID <tid>.

MYQ,q,jobs,TID=tid,TOT,mf

The five parameters are optional and may appear in any order.

<q> is the desired queue:

- I - input
- O - output
- P - punch
- E - execute
- S - special (film, plot, etc.)
- J - Janus (jobs currently being read, printed, punched at Central Site)
- A - all queues
- omitted - same as A

<jobs> is one of:

- ALL - list all jobs (overrides TID=tid)
- omitted - list only those jobs belonging to the user's (requestor's) terminal id or that specified by TID=tid

TID=tid Restrict the list to those jobs belonging to the specified <tid> (use TID=000 for Central Site) (if omitted, all TID's are searched (see <jobs>))

TOT List only the totals for desired queues (if omitted, individual jobs are listed with statistics related to each queue)

<mf> is a specific mainframe:

- MFE - CDC CYBER 750
- MFF - CDC CYBER 176
- omitted - both MFE and MFF

** FIND (or J) Command **

FIND,jobstem

Search all queues for those jobs beginning with <jobstem>
(the first 1-7 characters of a job name).

FIND,jobstem,q,mf

<q> and <mf> are as described above.

Note that <jobstem> is required and must be the first
parameter.

** Examples **

MYQ,0 List output jobs belonging to the user.

MYQ,ALL List all jobs in all queues.

MYQ,MFF List all jobs (176) belonging to the user.

MYQ,TID=142,0

List all output jobs belonging to remote batch terminal
142.

MYQ,TOT,ALL List totals for all queues.

MYQ,TOT List totals of jobs belonging to the user.

FIND,QYR List all jobs beginning with QYR.

FIND,xxxx,E,MFE

List all executing jobs beginning with 'xxxx' on the 750.

FIND,ABCDE4H List job 'ABCDE4H' information.

FIND,I,E List all executing jobs beginning with I.

FIND Illegal. <jobstem> is required.

** Error Messages **

MYQ - QAC ERROR NUMBER # - # (same for FIND or J)

MYQ - JAC ERROR NUMBER # - # (same for FIND or J)

MYQ - INTERNAL ERROR NUMBER # (same for FIND or J)

System error. Call User Services (202) 227-1907.

SYNTAX ERROR IN PARAMETER # OF MYQ CALL (same for FIND or J)

User error. Correct and re-type command.

** Q Command **

Q List number of jobs in each queue.

Q,jobname,q

<jobname> is the first 3-7 characters of the job name.
<q> is as described above, except S has a different meaning (see below).

Q,S

Q,SYNTAX Display Q command syntax.

** Examples **

Q,xxxx List all jobs submitted by user <xxxx>.

Q,xxxx,I List all jobs submitted by user <xxxx> which are in the input queue.

Q,IF6E List all routed jobs of user F6E.

Q,IF6E,O List all routed output jobs of user F6E.

*** Messages Between Consoles ***

** From Central Site Operator **

These messages may not be avoided by the command LOCK,ON. They will print immediately even if a program is in execution, then the run will resume. All messages become part of the dayfile record. All messages from the operator are preceded by the time the message was sent (see example at bottom of page).

Certain frequent message patterns are:

***, PLS LOGOUT IN xx MIN

***, PLS LOGOUT GOING DOWN

Duration of interruption usually longer than 15 minutes.
Try to save or catalog files and LOGOUT as soon as possible.

***, PLS LOGOUT NOW

Intercom must be dropped.

***, INTERCOM OFF IN xx MIN

Approaching the end of the shift in which Intercom is available. Begin to catalog files and LOGOUT within xx minutes.

***, PLS PURGE UNNEEDED FILES

***, PLS RETURN UNNEEDED FILES

Disk file space is filled. Return local files that are no longer required and/or purge (and return) some files.

***, MACHINE IN STEP FOR xx MIN

***, SLOW RESPONSE FOR xx MIN

Response may be very slow or seem nonexistent for a few minutes. Deadstart is not yet planned. Be patient for 2 to 10 minutes.

If, instead of ***, the message contains your terminal id, it is not a broadcast message, but is specifically directed to you. Please acknowledge or LOGOUT at once, if requested.

14.26.57.555, OK <-- response to user 555 after a message is received

**** To Central Site Operator ****

M, <message text>

M or MESSAGE will send a message to the Central Site operator from command or EDITOR mode only if no other message is pending at the Central Site console. If the console is busy, you must retype the command when it is no longer busy. However, the Central Site operator should not be bothered. If you need any help, call User Services or type SEND,USERSERVIC (see below). All messages become part of the system dayfile. If you must use this command, do not use any special characters listed on page A-2 of this manual. For example,

M, please evict job xxxnnn from INP Q

**** To Another Terminal ****

SEND commands may be directed only to logged-in intercom terminals who have not typed LOCK,ON (see page 4-20: LOCK, SITUATE). The entire 5- to 10-character user name must be typed in answer TO WHOM (not the 3-character terminal ID alone). If two or more users with the same user name are currently logged in, precede the user name with the 3-character terminal ID (e.g., SEND,555-USERSERVIC). A conversation may be carried on during the SEND command and messages will be stacked in the buffer of the receiver unless locked. All messages are time stamped. It is not necessary to type END to receive a reply to the first part of a message. To terminate the SEND, END must not be followed by any character, not even a period.

Example: SEND,USERSERVIC
please call 555-1234 to help user
EDITOR problems with perm file
END

SEND, as described above, will prompt each user entry with 'TYPE MESSAGE OR END-'. To suppress this prompting, use 'SEND,xxxxyyyyyy,S', which will give an initial prompt of 'GO-' and then accept and send user messages until END is typed.

**** To a User Who is not Logged In ****

Messages may be sent to Intercom users who are not logged in by using BEGIN,SEND. The document describing SEND may be printed by:

BEGIN,DOCGET,,OTHER,,SEND,OUTPUT,MSACCES=<password>.

*** Saving Files ***

If you wish to keep a program or data file for use at another session, you must catalog it into the permanent file system.

SAVE lfn	(NETED)	<-- if <lfn> is a new
SAVE,lfn	(EDITOR)	<-- file name, it will be
		on a PF device
CATALOG,lfn,pfn,ID=xxxx,<parameters>.		<-- enter file in catalog
		(AC optional)

See also STORE command below.

*** STORE, FETCH, DISCARD Commands ***

Intercom commands STORE, FETCH, and DISCARD have been modified to be compatible with DTNSRDC accounting conventions. The job order number corresponding to the LOGIN access number will be the AC to catalog the file. If the xxxx 4-character ID for the file is omitted, it is taken from the LOGIN user ID.

STORE,lfn
STORE,lfn,xxxx is the equivalent of:

CATALOG,lfn,ID=xxxx,AC=jjjjjjjjjj.

where jjjjjjjjjj is the job order number corresponding to the LOGIN access number.

The STORE command will put the file on a *PF device if it is not initially. As with a CATALOG, a 3-line message will be typed if the STORE is successful. The STORE command has no provision for passwords.

FETCH,lfn
FETCH,lfn,xxxx is the equivalent of:

ATTACH,lfn,ID=xxxx.

No message will be typed if the FETCH is successful.

DISCARD,lfn
DISCARD,lfn,xxxx is the equivalent of:

ATTACH,lfn,ID=xxxx. <-- if not already attached
PURGE,lfn.
RETURN,lfn.

If <lfn> was successfully purged, a 3-line message will be typed.

*** Additional Intercom Commands ***

ASSETS Display the user's terminal status.

DAY,FLUSH Force printing of dayfile messages at the terminal.
(Useful in procedures to clear the buffer so that system
messages are not mixed in with program output.)

DAY,OFF Suppress printing of dayfile messages at the terminal.

DAY,ON Print dayfile messages at the terminal. (default)

EFL,xxxxxx Change execution field length to xxxxxx (up to 200000).
EFL,0 The second and third forms are used to restore the default
EFL field length (65000). The higher the EFL, the slower the
 response.

ETL,xxx Change the command time limit to xxx octal seconds
ETL,0 (normally up to 500 (320 decimal)). The second and third
ETL forms are used to restore the default command time limit
 of 24 (20 decimal) seconds on MFF, 74 (60 decimal) seconds
 on MFE. A short time limit is useful in debugging a
 looping program to minimize charges. When the time limit
 has been reached, the message 'CP TIME LIMIT' will be
 typed.

ERRORS,compiler
ERRORS,compiler,SUP Reads compilation listing on file OUTPUT and lists the
 statements in error with the diagnostic messages.
 <compiler> is one of: ALGOL (or AL), COBOL (or COB),
 COMPASS (or COM) or FTN (or F). SUP (or S) will suppress
 informative/trivial error messages.

FILES List local, input, executing, output and punch files. If
 lfn is preceded by *, it is an attached PF. If lfn is
 preceded by \$, it is connected to the terminal. See
 CLEAR command (page 5-7) for a quick way to return all
 files.

LOCK,OFF After this command, you can again receive messages.

LOCK,ON After this command, you cannot receive messages from other logged-in terminals (see page 4-17: SEND). Messages from Central Site cannot be inhibited.

SCREEN,xxx,yy

SCREEN,xxx When a program prints output on an Intercom terminal, the system generates a 'go to next line' after each 72 characters even if the output is longer. On wide carriage terminals, it is possible to print up to 132 characters per line by typing 'SCREEN,132'. The number of characters per line is switched to 132. Any number less than 132 may be used, such as 'SCREEN,120'. To reset from long lines back to default, type 'SCREEN'. A user at a 72-character/line terminal device should not use the command since all characters after 72 would overprint the last character at the end of the line. <yy> specifies the number of vertical lines on a display terminal. It is ignored for teletypes and similar terminals. (See page 18-21 for an example of SCREEN,xxx,yy.)

SITUATE List the interactive and batch terminals currently logged in.

Interactive terminals are listed as

<tid>--<LOGIN name>

For example, 555-USERSERVIC. If preceded by an asterisk (*), that user typed LOCK,ON and cannot receive messages from other users.

Batch terminals are listed as

<tid>--<type>

For example, 142-200UT.

SITUATE,SHORT

SITUATE,S List the number of logged-in interactive terminals and batch terminals.

Some batch commands of interest include: CLEAR, RETAIN, REWALL, SUMMARY. See Chapter 5.

*** XEQ Command ***

The Intercom XEQ command may be used to execute programs. It (or a CCL procedure (see chapter 6)) is required to make use of certain loader commands: EXECUTE, LDSET, LIBLOAD, LOAD, NOGO, SLOAD, SATISFY. (see page 8-4,5)

To initiate program loading, enter:

XEQ

The system responds:

OPTION=

Enter one of the options (loader commands). If EXECUTE, NOGO or a filename is not entered, the system continues to request OPTION= entries. To leave the XEQ command, respond to the OPTION= request with END.

If the load sequence ends with EXECUTE, NOGO or a filename, the short form:

XEQ,option1,option2,...,optionn

may be used.

When executing a program by filename (e.g., 'LGO'), either 'LGO' or 'XEQ,LGO' or 'XEQ,LOAD=LGO' may be used.

OPTIONS= entries are entered as follows:

EXECUTE=ename,param1,param2,...,paramn
EXECUTE=,param1,param2,...,paramn

NOGO=name
NOGO

name,param1,param2,...,paramn <-- not in single line XEQ

LOAD=lf1,lf2,...,lfnn

LDSET,option1,option2,...,optionn
LDSET=option1,option2,...,optionn

FILES=lf1/lf2/.../lfnn

For additional information on the various options, see pages 8-4,5; INT, 3-29.

** XEQ Examples **

1. Program uses subroutines on libraries NSRDC5 and NSRDC:

```
ATTACH,NSRDC5.  
ATTACH,NSRDC.  
FTN5,I=prog,L=out.  
XEQ,LDSET,LIB=NSRDC5/NSRDC,LOAD=LGO
```

Another way, without using XEQ, is:

```
FTN5,I=prog,L=out  
ATTACH,NSRDC5  
ATTACH,NSRDC  
LIBRARY,NSRDC5,NSRDC  
LGO
```

2. Main program uses all subroutines on files SUBS and MORSUBS.

Main program is in EDITOR working storage.

```
RUN,FTN5,N  
XEQ,LOAD=LGO,SUBS,MORSUBS,EXECUTE
```

3. You have a program written for another machine which presets memory to zero before starting execution. A full map with cross reference is desired for offline printing. The program is in file LGO.

```
DISCONT,OUTPUT  
XEQ,LDSET=MAP=SBX,PRESET=ZERO,LOAD=LGO  
ROUTE,OUTPUT,DC=PR,TID=C,FID=*xxxx
```

4. Same as 3, only override the first PROGRAM statement file name with the file name TAPE14.

```
DISCONT,OUTPUT  
XEQ,LDSET,MAP=SBX,PRESET=ZERO,LOAD=LGO,EXECUTE=,TAPE14
```

5. Create a core image (or absolute) module (CIM) from the relocatable module(s) on file LGO

```
XEQ,LOAD=LGO,NOGO=CIM
```

The core image module, CIM, may now be used to test the program using several sets of test data by using a command such as one of the following:

```
CIM  
CIM,TEST1  
CIM,TEST2
```

where TEST1 and TEST2 are files containing test data and override the first PROGRAM statement file name. This will save lengthy loader processing since the loader will not have to find all the required subprograms for each load. If the program is found to work, CIM can then be cataloged (provided a 'REQUEST,CIM,*PF' preceded the XEQ).

*** PAGE ***

Local files may be examined using the PAGE command (see INT, 6-1). PAGE provides:

- positioning forward or backward within the file
- tabs for scanning long lines
- testing for the presence or absence of a character string
- transferring selected lines to a file for later printing
- exit and re-entry without loss of parameter information or the line index file.

PAGE is generally faster than the CDC EDITOR for looking at an output file.

PAGE accesses NOS/BE files with z-type records. Each unit record (line) is up to 150 characters. A longer line will be split into lines of 150 characters or less. A display-page is 10 lines.

The following control statement will initialize the PAGE program:

```
PAGE,lfn1,lfn2      where <lfn1> is the file to be paged
                      (default: OUTPUT)
                      <lfn2> is the print file
                      (default: PRINT)
```

PAGE will respond with:

READY..

after which any number of PAGE commands are entered in one line, separated by commas. These commands are executed consecutively. If more than one display command is given, only the final display-page is typed. To leave PAGE, type E or Q.

The last line of each display-page is 'line nnnnn', where nnnnn is the PAGE line number of the first line typed.

If an error is found in a command (sequence), the current display-page is typed, followed by 'u/xxx...' , where xxx... is the unprocessed portion of the command (sequence).

If the user aborts (%A) a search or print command, control remains in PAGE.

** HELP Command **

PAGE includes several lists which explain the various commands. Enter HELP for a description of the lists available.

** General Control Commands **

R Repeat the previous command line.

E or Q Exit the PAGE program.

** Display Format Commands **

Tnn Display line beginning at character position nn
(initially set to T1)

S Display the first 72 characters of the line.
(S mode is default)
To see more of the line, enter T73. To restore to the
first 72 characters, enter T1.

F Display each non-blank line in its entirety, using several
display lines if necessary.

** Line Location and Page Searching Commands **

nn Go to line <nn>.
(Display 10 lines starting at line nn.)

+nn Go forward <nn> lines....
(Display 10 lines beginning nn lines after start of last
display-page.)

-nn Go back <nn> lines.
(Display 10 lines beginning nn lines before start of last
display-page.)

+ Go forward one display-page.
(Display next 10 lines.)

- Go back one display-page.
(Display previous 10 lines.)

* Go forward to end-of-file.

H or +H Go forward to next header line.
(A header line has a '1' in column one.)

-H Go back to last header line.

** String Searching Commands **

These commands search forward or backward for the presence or absence of a specified character string. The first line satisfying the search condition is displayed as the first line of a page of information. If a search condition is not met, a message states that the beginning or end of the file has been reached.

String search commands have the following form:

direction	type	range	condition	string
where				
direction				
+ or omitted	Search forward.			
-	Search backward.			
type				
omitted	Search every line.			
H	Search header lines.			
range				
omitted	String is anywhere in the line.			
(i)	String must start in column i.			
(i-j)	String must start somewhere between columns i and j, inclusive.			
condition				
=	String must be in the line.			
"	String must not be in the line.			
string				
dxxx...d	d is the delimiter and is any character not in the search string xxx... .			

Examples of string searching commands:

=/ABCD/	Search forward for the first occurrence of ABCD anywhere in a line and list 10 lines
0,H"/FTN/	Go to beginning of file, then search forward for the first header line which does not contain 'FTN'. (0, is not needed if already at start of file.)
-(12)=*SEVERITY*	search backward for the first line which has SEVERITY starting in column 12. This can be used to locate FTN error diagnostics in a compilation listing. To search forward, remove the minus (-).
-H(1-9)=*VER*	Search backward for first header line which has VER starting somewhere in columns 1-9, inclusive.

*** Calculator Mode ***

** Using CCL **

Certain CYBER Control Language (CCL) verbs provide the user with a limited (integer only) calculator mode capability. They may be used in batch jobs but are most useful interactively.

DISPLAY,exp.

Evaluate the expression <exp> and display as a decimal integer with its octal equivalent. Note that if <exp> ends in a right parenthesis, you must add a terminator.

SET,R1=exp.

R2 Set any of the 4 CCL registers (3 local and 1 global) to
R3 an expression. The registers may then be used in the
R1G expression in a DISPLAY statement.

DISPLAY and SET are described more fully in NOSBE, Chap 5, and in the document describing CCL:

BEGIN,DOCGET,,OTHER,,CCL,OUTPUT,MSACCES=<pw>.

** Using FORTRAN **

A simple FORTRAN 4 program can be created (because the FTN4 compiler will supply the missing PROGRAM and END statements) to approximate a calculator mode.

with NETED

with EDITOR

ATTACH,NETED,NETEDF
NETED,T
;PRINT *, <expression>
.
SAVE
BEGIN,RUNFTN,,T

EDITOR
CREATE,SUP -or- C,S
;PRINT *, <expression>
=
RUN,F

If you use FTN5, you must add ';END' after the PRINT statement and use 'BEGIN,RUNFTN5,,T' or 'RUN,FTN5'. This will also display the warning message for the missing PROGRAM statement.

*** ASCII and Intercom ***

The default character set for CDC NOS/BE (including Intercom) is the 63-character set shown in Appendix A. A package of subprograms is available in library NSRDC (see page 18-7) to output ASCII to an interactive terminal. In addition to providing upper and lower case letters, control sequences recognized by various "smart" terminals can also be sent. Diablo 630, Omron, Tektronix and VT100 terminals are currently supported. If you need assistance in developing support for other terminals, contact User Services.

The ASCII support package consists of a procedure (ASCII0) and ten subprograms (ASCII, ASCIIII, D630I, OMRONI, TEKTRI, VT100I, ASCADD, ASCADM, ASCBSX, ASCGET, ASCLN, ASCPUT, ASCTXT) which perform the following functions:

ASCII0 - generate labelled common blocks /ASCII/, /D630/,
/OMRON/, /TEKTRN/, and /VT100/

ASCII - create an ASCII message

ASCIIII - initialize /ASCII /

D630I - initialize /D630I /

OMRONI - initialize /OMRON /

TEKTRI - initialize /TEKTRN/

VT100I - initialize /VT100 /

ASCADD - add an ASCII string

ASCADM - add an ASCII string multiple times

ASCBSX - remove backspaces and ctrl-X

ASCGET - get an ASCII character

ASCLN - get length of an ASCII string

ASCPUT - put an ASCII character

ASCTXT - convert Display Code string to ASCII

A general document describing the ASCII package may be printed by:

BEGIN,DOCGET,,OTHER,,ASCII,OUTPUT,MSACCES=<password>.

Documents describing the routines may be printed by:

BEGIN,DOCGET,,PROCFIL,,ASCII0,OUTPUT,MSACCES=<password>.

BEGIN,DOCGET,,NSRDC,,rtn,OUTPUT,MSACCES=<password>.

where <rtn> is ASCII, ASCIIII, D630I, OMRONI, TEKTRI, VT100I, ASCADD, ASCADM, ASCBSX, ASCGET, ASCLN, ASCPUT, or ASCTXT.

To print all the above documents, use:

BEGIN,ASCDoc,,OUTPUT,fid,MSACCES=<password>.

where <fid> is the file ID to use to route the output to print on the Xerox 8700. If <fid> is omitted, the output file is not routed.

*** Intercom Backup Decks ***

To create a backup deck at Central Site from an Intercom source code permanent file through interactive input:

Punch at Central Site:

```
ATTACH,TEMP,pfn,ID=xxxx.      <-- get file to be punched
BATCH,TEMP,PUNCH,xxxx        <-- deck banner card has 'Ixxxx00'
-or-
ROUTE,TEMP,DC=PU,TID=C,FID=*xxxx. <-- banner card has 'xxxx0nn'
```

In both cases, the permanent file is copied before routing and the 'nn' on the banner card is assigned by the system.

If accidentally routed to your own punch queue:

```
FILES          <-- to get <lfn> from punch queue list
BATCH,<lfn>,LOCAL
ROUTE,<lfn>,DC=PU,TID=C,FID=*xxxx.
-or-
BATCH,<lfn>,PUNCH,xxxx
```

Note: If <lfn> is to be retained on permanent file, copy it to a permanent file (*PF) device first, then catalog the copy.

These examples will cause the cards to be punched in 026 code. For 029 code, use

```
ROUTE,<lfn>,DC=PU,EC=029,TID=C,FID=*xxxx.
```

*** External Storage Media ***

** Paper Tape -- Cassette **

To use paper tape for input to a running program, each line to be read must terminate with carriage return, line feed, three RUBOUTs, and X-OFF (hold <ctrl> key while typing S), then three or more RUBOUTs. If a command to initiate execution of the program is the first thing on the paper tape, the tape may be readied before typing TAPE,ON. Otherwise, leave the tape in free or stop position, type the initiate execution command (e.g., NETED,NEWFIL) followed by carriage return (CR), line feed (LF), X-OFF. Then place the tape in auto mode, or wait for first input request, and start.

To read previously prepared paper tape under EDITOR, ready the tape in the reader, in the auto position, type TAPE,ON. The system will respond with .. and start to read the tape. If the tape does not start, use the start lever. When a paper tape reaches the ending RUBOUTs, it will stop. It may be manually stopped before that or may contain the ending commands. To exit from the paper tape mode, type TAPE,OFF followed by CR, LF and X-OFF.

When in the TAPE,ON mode any command typed in on the console must be followed by CR, LF, and X-OFF before it will be accepted by the system.

Paper tape for EDITOR input may be prepared with line numbers on, i.e., 100= ;PROGRAM SHORT(...) The command sequence is:

EDITOR

D,A to clear edit file

TAPE,ON and allow the tape to read in immediately

When paper tape for EDITOR input does not contain any line numbers, (i.e., ;PROGRAM DIFF(...)) The command sequence is:

EDITOR

TAPE,ON paper tape must not be ready in the auto position.

C,S CR LF X-OFF for line counter. Then start the paper tape reading after system responds. Tape will read to end supplying automatic line numbers.

= This character alone on a line or as last line on paper tape. If typed in, must have CR, LF, X-OFF.

TAPE,OFF CR LF X-OFF

L,A List whole program to check system assigned line numbers.

After terminating tape read with TAPE,OFF, if the system does not seem to accept commands, then type X-OFF and repeat TAPE,OFF sequence.

***** JOB PROCESSING CONTROL STATEMENTS *****

*** Introduction ***

1. For card decks, the following job card requirements apply at the DTNSRDC site:

MFF - CDC CYBER 176 - orange	(Carderock)
yellow stripe	(Annapolis) *
MFE - CDC CYBER 750 - blue or purple	(Carderock)
yellow stripe	(Annapolis) *

No other cards in the deck should be any of the above colors. At the DTNSRDC site, your name and code must be on the job card as comments.

Some remote sites have other standards for color of cards used or for name and code information placement on the job card.

2. The second statement must be the CHARGE card. Other control statements follow.

3. All control statements begin in column 1 and, unless otherwise indicated, have no embedded blanks. A period or close parenthesis must terminate the control statement parameters. Comments may follow the command terminator. Each control statement is a request for the system to load and execute a program of that name.

4. On the CDC NOS/BE system, all control statements are placed in the first record (that is, the sequence of control statements is terminated by an end-of-record, which, for a card deck, contains a 7/8/9 punch in column 1). This is represented in the examples by <eor>.

Subsequent records are data which the control statements must process.

A green end-of-information card (6/7/8/9 punch in column 1) must be a last card of the card deck. The Operations staff should add one, if it is missing. There is no special EOI indicator for jobs created interactively. EOI is represented in the examples by <eoi>.

No characters listed on page A-2 (Display Code greater than 57) should be used on any control statement or message to the operator.

* - indicate the mainframe on the Job Request Card accompanying the job.

*** Control Statement Formats ***

A control statement has one of the following forms (see page ii: notation):

examples

prog.	REDUCE, SUMMARY
prog,<positional-parameters>.	CHARGE, SKIPF
prog,<keyword-parameters>.	AUDIT, LDSET, UPDATE
prog,<positional,keyword>.	CATALOG, FILE, ROUTE
prog,<list>.	RETURN, REWIND, SYSBULL, UNLOAD

prog is the name of the program or command.

<positional-parameters> is a string whose entries must be in a specific order. Omitted parameters are indicated by 2 successive commas.

<keyword-parameters> is a string whose entries are identified by name or name=value (e.g., AI=I, ID=xxxx) and may be in any order.

<positional,keyword> is a string of both positional and keyword parameters. When a control statement has any positional parameters, they must appear before any keyword parameters.

<list> is a string of unrelated parameters whose position is unimportant.

Most control statements may not be continued. Those which may (e.g., ATTACH, BEGIN, CATALOG) are continued by ending the first line/card after a complete parameter (including the comma) and beginning the next parameter on the next line/card. To continue a control statement at an interactive terminal, a CCL procedure must be created and executed (see chapter 6).

Parameters which contain non-alphanumeric characters must, in most cases, be defined as literals. A literal is a \$-delimited string, that is, a string of characters preceded and followed by a dollar sign (\$<string>\$). If the string contains a \$, it is represented by \$\$, thus, TAN\$ must be defined as \$TAN\$\$\$.

*** Job Card ***

The job card is the first statement of the job and defines the job name and machine requirements as well as the punch mode (when the job is a card deck). It has the form:

jobname,CM65000,T60,I077777,MT1,GE1,PE1,HD1,P3,
RP1,Dzznn,STmfx. name/code kp

jobname a 4- to 7-character name of the form xxxxyyy
 xxxx - user's registered initials
 (contact code 189.3 to register as a user)
 yyy - selected by the user

Characters 6 and 7 of the job name are modified by the NOS/BE system.

Note: Some remote installations may alter the format of the jobname and/or comments field (99 name).

Other Parameters

In the fields below, the identifying letters must prefix the numbers. Unneeded fields may be omitted, but there may be no embedded blanks. Parameters following blanks are ignored and default values are used. (See page 11-12 for tape density parameters.)

CM maximum central memory (octal). Don't use unless >65000 required.
(default: 65000; max: 377700)

T total job time in decimal seconds (at each mainframe's CPU rate).
(default: 60 (MFF), 120 (MFE); maximum: 32766)

IO total I/O time in decimal seconds. (default: infinity)

MT number of 7-track magnetic tape drives.
(max: 2)

GE number of 9-track, 6250 bpi magnetic tape drives.
(max: 2)

PE number of 9-track, 1600 bpi magnetic tape drives.
(max: 3)

HD number of 9-track, 800 bpi magnetic tape drives.
(max: 2)

NT same as DTNSRDC default. Don't use -- specify particular type (GE, PE, HD).

Note: MT+GE+PE+HD+NT may not exceed 2 for P3 or P4 jobs.

P priority of job
4 - prime shift - express
3 - prime shift - regular (default)
2 - deferred - jobs will run during prime shift only if the
higher priority workload is light
1 - deferred - jobs will not start until after prime shift
(same rates as P2)
0 - block time - jobs will be processed after regular batch
workload (see page 3-8)
emergency - by special written request only, see POLICY
(See POLICY for CM, time, tape and user device set combinations
for P3, P4.)

RP number of device set drives. (This parameter is ignored. It is
listed here for information.)
(max: 1 at a time in non-block time; 2 in block time; none for P4)

D job dependency (see page 5-10: TRANSF; NOSBE, 4-86)

STmfx Site-id (MFF, MFE) where job is to run.

name your name for identification of the output, beginning about
column 50.

code your organization code.

kp columns 79-80
omitted or 26 - control cards (or entire job) punched in 026 mode
29 - control cards (or entire job) punched in 029 mode

** Default Job Statement **

xxxx. implies
xxxx000,CM65000,T60,I077777,MT0,GEO,PEO,HD0,P3,RPO. <-- MFF
xxxx000,CM65000,T120,I077777,MT0,GEO,PEO,HD0,P3,RPO. <-- MFE

Jobs waiting to be run are divided into groups according to required hardware (tapes) and priority (P). CM and T parameters on the job card specify maximum central memory in octal and central processor time in decimal seconds for each job. These parameters along with P determine when a job may be selected for execution. If the CM or T parameters exceed the allowed size for a given P group the P is reduced. P4 jobs in the input queue are initiated before any P3.

Within priority groups, jobs with low CM and low T are selected first since each priority group is subdivided. Memory subdivisions occur 40000, 100000, 150000, 220000 and 300000. Time subdivisions occur at 40, 100 and 200 seconds.

See page 7-5 for the additional parameters needed to process jobs for classified projects.

*** CHARGE Card ***

For each job, an accounting routine validates the user and computer access number to verify that there are funds available and the user is authorized to utilize those funds. To obtain a computer access number, a DTNSRDC job order number must be registered with or obtained from code 189.3, (202) 227-1910.

The CHARGE card is the second statement in the job and has the form:

CHARGE,xxxx,<accessnmbr>.

The fields following the word charge are positional.

xxxx Your DTNSRDC registered user initials (e.g.,
 AMDS, CASG).

<accessnmbr> Ten characters of computer access number.

*** WARNING ***

The WARNING statement should be used to identify jobs with output meant for limited distribution.

WARNING,type.
WARNING,type,lfn.

where type indicates the desired banner as follows:

Type	Banner
FOUO or OFFICIAL	FOR OFFICIAL USE ONLY
UNCLASS	UNCLASSIFIED
PRIVACY	PERSONAL DATA PRIVACY ACT OF 1974
CONFIDENTIAL	CONFIDENTIAL
SECRET	SECRET

When <lfn> is omitted, the banner page is written to file OUTPUT and a series of messages is written in the user's dayfile denoting the <type> of the job. When <lfn> is specified, the banner is written to that file with no dayfile messages.

Any errors in the control statement will abort the job. WARNING should follow the CHARGE card. CONFIDENTIAL and SECRET may be used only when running during classified time on the classified machine.

** Examples **

1. Output file contains personal data:

```
jobname,....
CHARGE,....
WARNING,PRIVACY.
...
<eoi>
```

2. Confidential file CFILE is to be ROUTED for printing:

```
jobname,...,CONFIDENTIAL,CFILE.          <-- banner at start
CHARGE,....
... any commands not using CFILE may occur here
ATTACH,MYCONFIDENTIALFILE,ID=xxxx,PW=read.
COPYF,MYCONFI,CFILE.
WARNING,CONFIDENTIAL,CFILE.              <-- banner at end
ROUTE,CFILE,DC=PR,TID=C,FC=1C.
...
<eoi>
```

(See page 7-5 for the job card security parameters.)

*** Other NOS/BE Control Statements ***

- CLEAR. Unload all files except INPUT and OUTPUT.
To unload INPUT and OUTPUT, they must be specifically listed. Note that INPUT will be ignored in batch. (See page 5-8: RETAIN) (DTNSRDC addition to NOS/BE)
- COMMENT. For the insertion of comments into the control statement record. It will be displayed in dayfile at operator's console and at the Intercom terminal, if executing interactively. The console does not flash for operator action.
- DMP,111111.
DMP,ffffff,111111.
Dump memory from relative octal address fffffff thru 111111. If fffffff is omitted, 0 is used. The dump will stop at the current field length (FL).
- EXIT. When there is an abnormal end to program execution, the control statements following the EXIT statement will be executed. For example, EXIT.
DMP(0,25000)
- EXIT,S. If you wish to execute additional statements after an abnormal ending during compilation or simple control statements having illegal positional parameters, these statements must follow an 'EXIT,S.'. Compiler errors will not cause a branch to 'EXIT.'. Any error which branches to 'EXIT.' will execute 'EXIT,S.' if it is encountered first.
- EXIT,U. If you wish to execute additional statements regardless of whether the job ran or not, these statements must follow an 'EXIT,U.'.

LIMIT,size. Change the amount of mass storage (disk space) which may be used at one time during the job. Size is the (octal) number of blocks of 64 PRU's (4096 words). The default mass storage limit is 1600 octal blocks (57,344 PRU's). The maximum is 7777 octal blocks or 777,700 octal (262,080) PRU's (LIMIT,7777.). To keep a job from generating too much output, the limit may be set lower. The total allowed mass storage in use in a job includes input files, output files, newly created permanent files, and scratch files. RETURN or UNLOAD may be used to decrease the mass storage in use. (If you catalog a file or route to the output queue, the total mass storage is NOT decremented to account for the former scratch space.) INPUT files are eight to twenty card images per PRU, and OUTPUT files are five to ten lines per PRU. (see page 15-7)

MODE,n. Mode error bypass should not be used at DTNSRDC. An attempt to ignore error mode 1 may cause an error mode 0.

NOTE(lfn)/line 1/line 2/.../line n

NOTE,lfn,r,/line 1/line 2/.../line n

Create a file of <n> lines. If 'r' is specified, the file is rewound before and after. '/' is a delimiter. It must immediately follow the terminator and may be any character. The default for <lfn> is OUTPUT. For a further description, including examples, use

BEGIN,DOCGET,,OTHER,,NOTE,OUTPUT,MSACCES=<password>.

This is a DTRSND C implementation of the NOS NOTE command.

PRNTSPY(lfn) Prints out the results of SPY. It reads the temporary file DOSSIER and the file <lfn> containing the relocatable binary program under scrutiny. If <lfn> is specified, the listed addresses are relative to each subprogram; if omitted, they are relative to RA. If multiple SPY runs are made in one job, 'RETURN,DOSSIER.' must be specified after each PRNTSPY to avoid duplicate output. (see page 13-23: SPY; CONV, Chapter 11)

REQUEST,lfn,*Q.

Put <lfn> on a queue device. It should be used before creating a file to be ROUTED (page 5-11). For other forms of the REQUEST statement, see pages 9-6, 9-14, 11-13.

RETAIN(lfn1,lfn2,...,lfn n)

RETAIN. Unload all files except the listed files, and INPUT and OUTPUT). (see page 5-7: CLEAR) (DTNSRDC addition to NOS/BE)

RETURN(1fn1,1fn2,...,1fnn)

RETURN(1fn) Rewind and unload tape and return tape drive to system for reassignment. It should be used as soon as tape usage is completed. The job card MT/GE/PE/HD/NT count is decreased.

To detach a disk file from user job, either RETURN or UNLOAD. (see page 5-10: UNLOAD)

Do not use both RETURN and UNLOAD on the same tape.

At end-of-job, all disk files are automatically returned; tapes are UNLOADED and RETURNED.

REWALL(1fn1,1fn2,...,1fnn)

REWALL. Rewind all files except listed files and INPUT and OUTPUT. (DTNSRDC addition to NOS/BE)

REWIND(1fn1,1fn2,...,1fnn)

REWIND(1fn)

Rewind disk or tape files.

SUMMARY. The accounting summary, up to the current point in the job, is put into your dayfile.

SYSBULL(blist)

Print the system bulletin(s) specified.

Three bulletins always exist:

INDEX - list of all bulletins currently available
(default if <blist> omitted)

BATCH - printed after the banner page of all batch jobs

LOGIN - typed at Intercom LOGIN time

BATCH and LOGIN may refer to other bulletins which you should print. 'SYSBULL,ALL.' will print all bulletins.

TRANSF,job1.

TRANSF,job1,...,jobn.

Subtract 1 from the dependency (D) counter for job <jobi> in the dependency set. If the counter is then 0, the job may execute. (See page 5-4: D parameter.) For example,

xxxxB and xxxxC depend on successful completion of
xxxxA
xxxxD depends on xxxxB and xxxxC

xxxxA,DVS00.

...

TRANSF(xxxxB,xxxxC)

...

<eoi>

xxxxB,DVS01.

...

TRANSF(xxxxD)

...

<eoi>

xxxxC,DVS01.

...

TRANSF(xxxxD)

...

<eoi>

xxxxD,DVS02.

...

<eoi>

Note that each job name, <jobi>, must be unique in the first 5 characters.

If possible, dependent jobs should be put into the input queue before the main job.

UNLOAD(lfn1,lfn2,...,lfnn)

UNLOAD(Lfn) Rewind and unload tape so operator may remove it from the tape drive. The job card MT/GE/PE/HD/NT count is not decreased, so another tape may be mounted. (See page 5-9: RETURN)

*** Describing a File ***

Each file accessed by Record Manager has an associated File Information Table (FIT) which defines the file and how it is accessed. The compilers define the FIT's required for the program and initialize them to default or user-supplied values. At execution time, this information may be changed by means of the FILE command. In this way, files created by a program written in one language may be read by one written in another language (e.g., files written by COBOL may be read by FORTRAN programs).

FILE,lfn,<parameters>.

where lfn is the file whose FIT is to be modified
<parameters> are the various fields of the FIT which may be user-defined. All are in keyword form (keyword=value).

A few of the keywords are:

BFS	buffer length in words (tape file)
BT	block type (C, K, I, ...)
CF	close file action (N, R, U)
CM	conversion mode (tape file)
EO	error option (tape parity errors) (A, ...)
ERL	maximum non-fatal error limit (default: 1)
FO	file organization (default: SQ)
KL	key length
LFN	replacement file name
MBL	maximum block length in characters
MRL/FL	maximum record length in characters
OF	open file action (N, R, E)
RB	number of records per block
RT	record type (Z, F, W, S, ...)

(See FTN5, F-2; FTN4, 16-6; BAM, 3-7)

Sample FILE Commands

FILE,PRT,BT=C,RT=Z,MRL=150.

Zero-byte terminated file for printing.

FILE,STRANG,RB=10,MRL=80,MBL=800,BT=K,RT=F,EO=A,ERL=25,BFS=512,CM=YES.

Stranger blocked coded tape. For K,F-type records, MBL must be a multiple of 10 and be greater than or equal to RB*MRL.

FILE,SIS,FO=IS,RT=F,KT=S,KL=10,MRL=100,RB=6.

Indexed sequential file.

FILE,INPUT,LFN=DATA. Substitute alternate input file for COBOL prog

A LDSET statement (page 8-6) is needed to make the information on the FILE statement available to a user program 'LDSET,FILES=lfn.' .

*** ROUTE ***

The ROUTE command is used for directing the disposition of a file and defining its characteristics.

The ROUTE disposition directives include selection of the terminal to process the output and specifying the forms on which the file is to be output. Note that with the ROUTE command, special forms (such as multiple-part paper) may not be specified at remote sites (see page 21-2 for PM). In addition, there are characteristics parameters (EC/IC) which define the external/internal coding and the print train or punch code to be used.

The ROUTE command has the form:

ROUTE, lfn, <parameters>.

The file being ROUTED must be on a queue device. If not, ROUTE will copy it before routing. For better performance, use a 'REQUEST, lfn, *Q' statement (page 5-8) before creating the file. All parameters are optional. In general, if a parameter is omitted, it retains the definition from the last ROUTE command which referenced that lfn (the exception is the DEF parameter).

** ROUTE Parameters **

terminal identification

TID	Return file to job origin.
TID=C	Output the file at Central Site.
TID=tid	ROUTE the file to Intercom Terminal with ID of <tid>.
absent	See note 1 on page 5-14.

deferred routing (batch only)

DEF	File disposition is deferred until end-of-job. DEF must be specified in each ROUTE command if end-of-job processing is intended. If used at an interactive terminal, the file is scratched (lost) at LOGOUT. DEF and DC=IN may not be used together.
absent	File disposition takes place immediately. The file is no longer available.

disposition code

DC=xx file description, where xx is one of:

- IN - place the file in the input queue
- PR - any printer
- PT - former high-speed remote on-line plotter
- PU - punch
- SC - evict (scratch) the file
- TA - make coded file on PDP 11/70 disk at NAVSEA
- TP - make binary file on PDP 11/70 disk at NAVSEA

The keywords FID and FC are ignored with no warning message when DC=IN. If DC=IN and DEF are both specified, the ROUTE command is not executed.

DC same as DC=SC

absent Same as previous reference to this lfn on a ROUTE command. If none, same as DC=SC, except for the following special file names:

special lfn	assumed DC	assumed EC
FILMPL	FL	
FILMPR	FR	
OUTPUT	PR	
PLOT	PT	
PUNCH	PU	026 (BCD - default)
PUNCH	PU	029 (when EC=029 used -- Central Site only)
PUNCHB	PU	SB (binary)
P80C	PU	80COL (80-column binary)

A disposition code need not be specified for a special file name. DC=SC may be used to prevent the file from being processed.

file identification (not needed for DC=IN)

FID The file name, while in the output queue, is the same as the job name.

FID=* As above, only the 6th and 7th positions are replaced by 2 unique sequence numbers. When FID=* is used, the name of the file sent is listed in the dayfile.

absent See note 1 on next page.

Example: assume filename = jobname = ABCDE3P.
assume next sequence number is 4G.
after FID -- ABCDE3P
after FID=* -- ABCDE4G

forms code (Central Site only) (see note 2 on next page)
FC=yy Use special-forms code for printer and punched output as given on page 5-15.

FC Use standard card or print forms.

absent See note 1 below.

site id (station identification)
ST=MFx MFx (x = E or F) is the mainframe where the file is to be sent. For DC=IN, is specifies where the job is to be run (overrides the ST parameter on the job card).

ST Process the file on the system where it originated.

absent See note 1 below.

repeat count (Central Site and 200-UT-type only)
REP=nn <nn> is number of extra copies for output files.
(default: 0; max: 31 (37B))

spacing code (CYBER 176 only)
SC=nn Spacing code for output sent to a 580 PFC printer.
nn is octal (0-77) and indicates the spacing code array.
The following spacing codes have been pre-defined:
SC description

0 default - same as SC=1
1 standard 6 lines per inch
2 standard 8 lines per inch
Do not use values which have not been pre-defined in the system. See note 2 below.

no complementary dayfile
NCD Used only when a 'ROUTE,OUTPUT,DEF,....' to a remote site has been used and the original job initiator does not want a copy of the dayfile at his site.

Note 1: When file identification, forms code, spacing code, terminal and/or station identification are omitted, it is the same as in the previous reference to the lfn on a ROUTE command. If omitted on the first/only ROUTE statement for an lfn, the defaults are FID, FC, SC, TID, and ST, respectively.

Note 2: On the CYBER 176, the specified forms code (FC) will invoke the corresponding spacing code (SC). If SC is specified in the ROUTE command, it overrides the implied value.

*** Special Forms Codes ***

For special print forms:

1P one-part unlined paper
* 2P-4P two-part to four-part plain
* 1L-4L one-part to four-part lined
1T one-part teletype (narrow) (<=72 columns)
3T three-part teletype (narrow) (<=72 columns)
BS report distribution envelopes
BL labels (2 across) supply
X3 std 1080 form (6-part)
X4 labels (3 across)

88 same as 1P at 8 lines per inch (200-UT only)
8T same as 1T at 8 lines per inch (200-UT only)

* 1C one-part confidential (CYBER 750 only)
* 1S one-part secret (CYBER 750 only)

** XH Xerox-8700 - 3-hole punched, duplex
** XI Xerox-8700 - 3-hole punched, simplex
** XL Xerox-8700 - unpunched
** XS Xerox-8700 - special requirements

* ED Finance form (earning and deduction)
PA Payroll form (four-part)
* W2 Finance form (IRS W-2)
* X1 Finance form (FICA)

For special punch card stock:

* BS Transaction cards
* 2H Annapolis transaction cards (clock)
NS Supply receipt
N1 Supply issue
* N2 Supply requisition
* N4 Milestone

For special purpose output:

WW microfiche (before using this, the requested
microfiche program must have been
cataloged by User Services.)

Additional special forms may be assigned if frequently used.

* - not stocked at Central Site

** - use procedure XEROX (see page 6-7 and CLIB/P)

*** ROUTE Examples ***

1. Send entire output and dayfile to another terminal (e.g., 012).

```
jobname,....  
CHARGE,....  
ROUTE,OUTPUT,DEF,DC=PR,TID=012.    <-- set to print at end of job  
...  
<eor>  
...  
<eoi>
```

Note that the file may be referenced before it is created. The routing information is saved until the file is sent to the (remote) output queue.

2. Put a job into the input queue. (This is similar to the Intercom BATCH command.)

```
jobname,....  
CHARGE,....  
REQUEST,FALCON,*Q.  
COPYE,INPUT,FALCON.  
ROUTE,FALCON,DC=IN.  
<eor>  
jobname,....    <-- complete job  
CHARGE,....    <-- to go into  
...            <-- input queue  
<eor>  
...  
<eoi>
```

3. Punch file PUNCH on special cards (e.g., Supply issue):

```
ROUTE,PUNCH,TID=C,FC=N1.    see page 5-15
```

Punch file ANY as an 029 deck:

```
ROUTE,ANY,DC=PU,EC=029,TID=C.    (Central Site)  
-or-  
BEGIN,CV029,,ANY,012.    (remote 1700 with terminal ID '012')
```

4. Send a copy of the output file to another terminal (e.g., 142).

```
jobname,....  
CHARGE,....  
...  
REWIND,OUTPUT.  
COPYE,OUTPUT,ANY.  
ROUTE,ANY,DC=PR,TID=142.  
<eor>  
...  
<eoi>
```

***** PROCEDURES (CCL) *****

*** Introduction ***

A procedure consists of a series of control statements residing on some file. Execution of a procedure is started by a BEGIN control statement. Return from a procedure is caused by a REVERT control statement within the procedure. A procedure may include nested calls to other procedures. A procedure may be interactive or non-interactive. Interactive procedures allow the user to request help for any parameter, as well as to define required parameters, parameter types, etc.

BEGIN and REVERT are a part of the CYBER Control Language (CCL) which also provides for skipping control statements, creating data records in procedures, parameter substitution in procedures, control statement duplication, and selective control statement execution.

The BEGIN statement, which may be continued on more than one line (except interactively), is used to transfer control to a procedure:

BEGIN,procnam,procfil,<params>.

procnam - the name of the procedure to be executed

procfil - the file containing the procedure <procnam>.

Generally, when <procfil> is omitted, the system public-access procedure file is attached automatically. A user procfil must be attached or otherwise created. It is recommended that the lfn for a user procfil not be PROCFIL.

<params> - zero or more parameters for procedure <procnam>.

REVERT is used to return from a procedure and is part of the procedure. If a REVERT statement is not found in the procedure, one is supplied at the end of the procedure by CCL. As in a regular control statement sequence, a fatal error during the execution of a statement in a procedure will cause the procedure statements to be skipped until an EXIT statement is encountered. A 'REVERT,ABORT' may be used to cause the calling control statement stream to abort also. For Example,

... (control statements to be executed)

REVERT.

EXIT. or EXIT(S)

... (any control statements to be executed if a fatal error occurs during the procedure)

REVERT,ABORT.

Libraries of procedures may be sequential files or EDITLIB user libraries (see chapter 17).

For a copy of our 42-page CCL document (condensed from NOSBE, Chapter 5, with many added examples) use:

BEGIN,DOCGET,,OTHER,,CCL,OUTPUT,MSACCES=<pw>.

*** Interactive Execution ***

1. For a list of the parameters: BEGIN,<pname>,<pfile>,<pi>?
2. For help with a parameter: BEGIN,<pname>,<pfile>,<pl>,...,<pi>?
3. For help with a parameter during interactive processing: <param>?
4. To accept the default value of a parameter during interactive execution: type an EOR (%eor) or an EOF (%eof).
5. To accept the default value for the current parameter and all remaining parameters: enter a terminator (period (.) or right parenthesis (')'). Note that if any required parameters remain undefined, they will continue to be requested until supplied.

*** Public-access Procedures - PROCFIL ***

The Computer Center maintains three libraries of procedures:

PROCFIL - all Center-supported procedures
I - interactive versions of a subset of PROCFIL
MSS - a subset of PROCFIL for use with the Mass Storage System

These procedure files are accessed only via the BEGIN statement. The second parameter of the BEGIN identifies the desired procedure file. When omitted, PROCFIL is used; when I or MSS, file I or MSS is used. You must not use PROCFIL, I or MSS as the lfn of any other file, or it will be used instead of the desired system file.

See CLIB for a list of the available procedures and CLIB/P for full documentation of each procedure.

Selected procedures are described in detail on the next pages.

AUDIT

Sorted user audit.

To use - BEGIN,AUDIT,,keyword-parameters.

keyword-parameters are those described on
page 9-5 (AI=, AC=, ID=, LF=, PF=)

Default - BEGIN,AUDIT,,AI=P,ID=xxxx,LF=OUTPUT. (batch)
BEGIN,AUDIT,,AI=I,ID=xxxx,LF=OUTPUT. (Intercom)

Example - BEGIN,AUDIT,,AI=P,ID=xxxx.

Max FL - 52000B

MSAUDIT

Sorted Mass Storage System (MSS) audit.

To use - BEGIN,MSAUDIT,MSS,lf,un,lo,day,pw.

lf - listable output
un - user id to be audited
(may not be used with LO=FP)
lo - list option (audit format)

One of:

S - short audit (length, mfn,
ct, m, # uses)
I - intermediate audit (S +
streams, create and
last access dates, pw,
ac, \$/day)

P - same as F

F - full MSAUDIT (3 lines/file)

FP - complete MSAUDIT of accesses
to all semi-private
files

day - OFF - Turn off dayfile messages for
individual MSAUDITs for LO=FP
(could generate wallpaper).
Has no effect on other
options.

ON - Turn on dayfile messages.

pw - PW=PW - for LO=F, list file
password, if any
omitted - do not list file passwords

Default - BEGIN,MSAUDIT,MSS,LF=OUTPUT,UN=xxxx,LO=F,DAY=ON.

Example - BEGIN,MSAUDIT,MSS,LO=I.

Max FL - 45000B

DOCGET

Get a document from a document file.

To use - BEGIN,DOCGET,,lib,id,doc,l,lo,copies,i,fid,
msacces,un.

- lib - Document file containing desired document(s). If permanent file, the full pfn is lib_DOCUMENTATION.
- id - ID under which document file is cataloged.
(Default: system public-access id)
- doc - Desired single document or
ALL - all documents or
omitted - multiple documents
(see 'i=')
- l - Listable output.
(Default: L=OUTPUT)
- lo - List option. One of:
LEFT or omitted - left-justified
anything else - centered on wide
paper (suitable
for microfiche)
(Default: LO=LEFT)
- copies - Number of copies. (use REP= on
ROUTE command, if possible)
(Default: COPIES=1)
- i - for multiple documents. This file
contains the names of the desired
documents, comma-separated.
(omit 'doc=' parameter)
- fid - Optional file ID to route output
file to the Xerox 8700. If
omitted, the extracted document(s)
will be in the file specified by l=.
- msacces - Your MSS access password. Required
only if you have not already
submitted it and if the document
file is on the Mass Storage System.
(All public-access document files
are on the MSS.)
- un - UN under which document file is
stored on the MSS.
(Default: UN=DOCU)

Example - For full document describing procedure DOCGET:
BEGIN,DOCGET,,PROCFIL,,DOCGET,MSACCES=<pw>.

GRIPE

Allow user to make gripes or suggestions directly to the computer.

To use - BEGIN,GRIPE,,output,prompt,whom,input.

output - Listable output, if user requests a copy of his comments. To have this print at the terminal, you must CONNECT,OUTPUT.

prompt - interactive use only
If NO - only minimal prompting is provided.
If omitted or anything else - full prompting is provided.

whom - Reserved for future use.

input - Batch: File containing gripe and administrative information.
Interactive: If gripe is to be entered directly from the keyboard, omit this parameter.
If the gripe was prepared earlier (e.g., using NETED), enter the lfn of the file containing the gripe (do not use lfn INPUT).

Default - Interactive: BEGIN,GRIPE,,OUTPUT.

Batch : BEGIN,GRIPE,,OUTPUT,,,INPUT.

Data - The gripe is entered in columns 1-70 of as many lines as required (please be brief) and is ended by a separate line containing only END.

For batch only, a line with administrative information for preparing a Trouble Form follows the gripe and contains:

col	contents
1-20	name
21-40	DTNSRDC code or company name
41-50	telephone
51-54	user initials (in the form xxxx)

Example - The simplest form for interactive use:

CONNECT,OUTPUT.
BEGIN,GRIPE.

The simplest form for batch use:

```

jobname,....
CHARGE,....
BEGIN,GRIPE.
<eor>
      (text of the gripe)
END
name   ...   code   ...   phone   init
<eoi>

```


RENAMAC Rename AC parameter (job order number) of all your permanent or MSS files.

To use - BEGIN,RENAMAC,,old,new,id,1,pw1,...,pw5.

- old - old AC (required)
- new - new AC (required)
- id - id of files to be renamed (req)
- 1 - listable output
 (default: OUTPUT)
- pw1-pw5 - up to 5 passwords for all files to
 be renamed
 (only TK, RD and CN passwords are
 required)

BEGIN,RENAMAC,,old,new,MSS,,pw1.

- old - old AC (optional)
 (if omitted, all your files are
 renamed)
- new - new AC (required)
- MSS - required to indicate that MSS files
 are to be renamed
- pw1 - MSACCES password, if not already
 submitted during the current job or
 interactive session.

Default - Permanent file rename, but no meaningful defaults.

Max FL - 35000B

UTILITY Execute a program on EDITLIB user library UTILITY.

To use - BEGIN,UTILITY,,prog,params.

- prog - program to be executed
- params - up to 19 parameters for 'prog'
 (keyword parameters must be
 \$-delimited)

Default - There is no meaningful default

Max FL - Depends on program being executed

XEROX

Route an output file to the Xerox 8700.

To use - BEGIN,XEROX,,lfn,fid,copies,job,paper,duplex,forms.

lfn - listable local file to be routed
fid - required file id for ROUTE command
copies - number of copies
job - Xerox 8700 jobname (LND=landscape;
PRT=portrait; cc=carriage control).

One of:

STDLND - cc+136x 64 - sans-serif
std printouts (compiles/
load maps, pgm output)

CMPLND - like STDLND, except each
subprogram begins on a
new page

LN8LND - cc+136x 86 - sans-serif
8 lpi (dayfiles, load
maps/compiles with PD=8)

2ACLND - cc+ 72x 64 - sans-serif
(twice)
two-across (like Center
Note)

STDPRN - cc+ 72x 64 - pica
1T-style (general
documentation)

ST3PRT - cc+ 94x 66 - sans-serif
(very large margins)

ST4PRT - cc+ 87x 64 - sans-serif
1T-style (general
documentation; manuals)

DOCPRN - like ST4PRT, except
documents have '.' in
column 76 to force each
to start on a new page

VAXDOC - like ST4PRT, except
each page is exactly 66
lines long with no cc

2UPPRT - cc+136x132 - sans-serif
(two-up compact storage
of standard printouts)

2U8PRT - cc+136x176 - sans-serif
(like 2UPPRT, 8 lpi)

paper - paper to be used. One of:

PLAIN - unpunched

anything else - 3-hole punched

duplex - omit for duplex (2-sided printing)

NO for simplex (1-sided printing)
(duplex should be used under normal
circumstances)

forms - Name of forms overlay.
omitted or NONE - no overlay
See XEROX document for available
forms overlays.

Default - BEGIN,XEROX,,LFN,*,1,STD LND,3HOLE,YES.
* - required parameter

Max FL - 13000B

***** SECURITY *****

This chapter discusses security procedures which pertain to the CDC computers at DTNSRDC.

The Privacy Act of 1974 is discussed on page 7-2.

Limited-distribution printouts are to be identified with a banner page containing 'FOR OFFICIAL USE ONLY', 'PERSONAL DATA PRIVACY ACT OF 1974', 'CONFIDENTIAL', or 'SECRET', as appropriate. Note that 'CONFIDENTIAL' and 'SECRET' may be used only during classified time on the CYBER 750. The job card and the WARNING command are used to generate these banner pages (see page 5-6).

Access numbers, assigned when valid job order numbers are registered, are used to access the computer via batch (CHARGE card) and Intercom (LOGIN process). These access numbers are provided to help protect against unauthorized charges to the user's account and should be treated as 'FOR OFFICIAL USE ONLY'. Access numbers are normally unrestricted but may be restricted to one or more users (contact Code 189.3, (202) 227-1910 to restrict access). Access numbers are changed at the start of each fiscal year and at other times during the year, if necessary.

To protect against unauthorized charges to the user's Intercom account, a User Turnkey Password must be defined by the user. This password should be changed frequently. See page 4-3.

Permanent files can be protected by passwords which control use of the file for everyone, including the owner. The turnkey (TK) password should be used to prohibit any unauthorized access to a file with sensitive data in it. The read-only (XR) password should be used to prohibit any unauthorized modification of a file. See page 9-8.

User device sets may be protected by a user-defined turnkey password, which, once defined, must be used in the MOUNT statement for that device set. See page 7-6.

Classified jobs must be run on the CYBER 750 during classified time.

For Mass Storage System (MSS) security, see page 10-1. Users should change the MSS access password frequently.

*** Privacy Act of 1974 ***

Public Law 93-579, which became effective in the fall of 1975, provides safeguards for individuals against invasion of their personal privacy. This law also provides serious individual penalties for violations. The following simple security rules and procedures are the responsibility of each user to follow to help maintain the integrity of the computer systems' files.

- 1) Users must have explicit permission before they access files, computer listings, cards, tapes, disk packs, documentation, etc., which belong to other individuals and projects. (Public access files as documented in this manual and in the Computer Center Notes (see page 21-1) are available to all users.)
- 2) Access numbers, job order numbers, user initials, tape ID's and Intercom login names must be used only when authorized.
- 3) Users have no authority to tamper with the operating system. Programs, routines and files (password, account, etc.), used for system maintenance may not be accessed.
- 4) Permanent files and tapes, containing personal data, should be protected with passwords (see page 9-8) and labels, respectively, and these identifications should be kept confidential and changed frequently to discourage disclosure of file data.
- 5) Intercom users should use the turnkey password (page 4-3) to protect their account numbers.
- 6) For assistance in password protection of your files or clarification of any of the above, contact User Services (202) 227-1907.
- 7) Users who require special handling for data tapes and output containing personal data should notify Operations Branch, Code 1896, (202) 227-1365.
- 8) Printed output containing data covered by the Privacy Act of 1974 should have 'PERSONAL DATA PRIVACY ACT OF 1974' printed on the top and bottom of each page.

See page 5-7, for a means of identifying printouts as private.

*** Classified Microfiche ***

Classified information may be put onto microfiche provided the following have been done.

- 1) "... the assigned security classification and abbreviated declassification instructions shall be conspicuously marked on the (microfiche) so as to be read by the unaided eye." *
- 2) "... these security markings shall also be included on each page captured on the (microfiche) so that when the image is retrieved, enlarged and displayed or printed, the security markings will be conspicuous and readable." *

To accomplish this,

- 1) User programs may have to be modified to print this information.
- 2) Compilation listings must be modified to include this information. Any text editor, such as NETED, may be used.

Note: The fiche will be returned with a strip of red tape backing the eye-readable line.

* OPNAVINST 5510.1F (which includes DOD 5200.1-R)

*** Processing Classified Jobs ***

Jobs for classified projects are run on the CDC CYBER 750 during classified time. Jobs at the same security level from one user are run concurrently.

** Physical Security **

Classified work must be delivered to the Dispatch counter in Building 17 at Carderock. The following measures will be enforced:

- The doors to the CYBER 750 room are locked.
- A doorbell will be used to call the operator on duty for service.
- The following items, externally marked with the proper classification, must be signed in or out in the log:
 - 1) card decks.
 - 2) magnetic tapes.
 - unclassified tapes may be in a slot or in Diebolds (certain CA and CB).
 - classified tapes must be stored in a safe (space is limited).
 - 3) output: cards, tapes or printout.

** Permanent Files **

The following restrictions apply to permanent files during classified time:

- Classified permanent files must be cataloged on user-owned classified device sets only.
- When the machine is in classified mode, unclassified permanent files must be cataloged only on user-owned device sets.
- Unclassified files may be read from the public permanent file set or the Mass Storage System.

**** Job Card ****

An additional parameter indicating the level of security must be included on every job card during classified time on the CYBER 750. There is no default; jobs without this parameter will be terminated.

This parameter indicates the level of security and causes a banner to be written on file OUTPUT. The job card may also specify that this banner be written on another file by putting the lfn as the next parameter. Note that the security and output lfn are normally the last parameter(s) before the terminator. If any job card parameters are placed after these, the output lfn must be specified. Caution: Do not choose an output lfn which starts the same as another job card parameter, or with the letters G, V or X.

See pages 5-3, 5-4, for other job card parameters.

*** Examples ***

jobname,CMnnnnn,Tnnn,CONFIDENTIAL.

This is a confidential job and a confidential banner is written on file OUTPUT.

jobname,CMnnnnn,Tnnn,SECRET,MYFILE.

This is a secret job and a secret banner is written on file MYFILE.

jobname,CMnnnnn,CONFIDE,Tnnn.

This is probably a bad job card. The confidential banner is written on file Tnnn. If Tnnn is a time limit, this job card must be:

jobname,CMnnnnn,Tnnn,CONFIDE.

*** Device Set Password ***

An additional level of security for device sets is available as a password on the MOUNT command. To activate this feature, the user must catalog on the pack a dummy file with pfn and ID of "PASSWORD" with a 1- to 9-character turnkey password (TK).

For example:

```
REQUEST,PASS,*PF,SN=setname.  
REWIND,PASS.  
CATALOG,PASS,PASSWORD,ID=PASSWORD,TK=<turnkey>.
```

Each MOUNT statement for that pack must then include the PW parameter

```
MOUNT,VSN=volser,SN=setname,PW=<turnkey>.
```

The device set owner may change this password by renaming the password file with a different turnkey.

***** LOADER *****

*** Introduction ***

The loader is responsible for loading all programs, resolving any external references, and optionally initiating execution.

Once loading of a program is started, no other control statements may interrupt the load sequence. For instance, a 'LOAD,lfn.' statement may only be followed by another 'LOAD,lfnl.' or one of the loader control statements (see pages 8-4,5,6,7) or MAP, REDUCE (see pages 8-8,9) or others listed in the Loader Manual.

A line at the end of the load map of a simple or overlay load contains 'xxxxxxB CM USED'. This is the minimum FL needed to load. A similar message in the dayfile also gives the program length. Use this information to save memory on future runs. At present, this information is not available with SEGLOAD.

*** Types of Loading ***

Loading differs according to whether the input is one or more object modules or a single core image module. Loading of object modules can involve overlay or segment generation and can result in one or more core image modules. A basic load results in one core image (absolute) module.

- | | |
|-----------------------|---|
| Object module loading | One or more object modules are loaded, libraries are searched for the external references, addresses are adjusted, and a core image module may be produced. |
| Core image loading | This is a special case because no external linkage or address adjustment is required. |
| Basic loading | All object code is loaded at the same time, resulting in a single core image module. |

Segmentation

For large programs, segmentation should be used to divide the program into several core image modules, called segments.

With segmentation, only those portions of the program needed at a given moment are in core. Different core image modules reside in the same area of memory at different times. Depending on execution requirements, different core image modules are loaded dynamically.

Features:

- Segmentation allows any number of levels, limited only to a total of 4093 segments.
- After segments have been generated, their loading is automatic.
- References between segments may be upward or downward.
- At execution time, a resident program is loaded which loads the root segment. Thereafter, it loads the other segments as required.

Overlay generation

Provided primarily for compatability with previous loaders.

Advantages of overlay over segmentation:

- 1 - Overlay core image modules can be handled by EDITLIB whereas segments cannot.

Disadvantages of overlay:

- 1 - Only 3 levels of overlay are permitted.
- 2 - User must insert calls into the program to load subsequent overlays.
- 3 - Restructuring of overlays requires source code modification. (Segmentation structure is defined outside the program, thus no source code modification is needed to restructure segments.)
- 4 - Each overlay must begin with a main program statement. (Segmentation has only one main program.)
- 5 - Only self and upward references are allowed.
- 6 - Entire loader must be in memory, whereas segmentation has only a small resident.

See page 8-14 ff for a sample of overlay in FORTRAN.

Overlay capsule generation

For large programs, overlay capsules may be used to divide the program into an absolute main program and one or more capsules which are loaded and unloaded by the user.

It is similar to segmentation in that there is one main program. It is similar to overlay in that the user must insert calls into the program to load, execute and unload capsules, and it can be handled by EDITLIB.

Overlay capsules use Common Memory Manager (CMM) and are, therefore, incompatible with the STATIC option of FTN5 and FTN4.

Advantages of capsules:

- 1 - The main overlay and capsules can be handled by EDITLIB with no special coding to determine the local file name of the library.
- 2 - Capsules can be brought in and released as the needs of the program change.
- 3 - There are no levels; any number of capsules can be brought in at once and capsules may load other capsules.

Disadvantages of capsules:

- 1 - User must insert calls into the program to load, execute and unload capsules.
- 2 - Restructuring of capsules requires source code modification.
- 3 - Capsules cannot refer to routines in other capsules, except by loading and executing a complete capsule, but may call routines in the main overlay.

CAUTION

Overlay capsules may have entry points other than the capsule name. If a program containing capsules is in an EDITLIB library, that library should not be seen by the loader. If it is, the loader may try to load a capsule instead of the desired subprogram. If you normally have such a library as a global library (page 8-8), you should clear it before loading and restore it after, if desired.

See page 8-16 ff for a sample of overlay capsules in FORTRAN.

*** Loader Control Statements ***

LOAD(lfn1,lfn2,...,lfnn)

Files, whose contents are to be loaded, are specified in one of the following forms:

- lfn - rewind (unless INPUT) before loading
- lfn/R - rewind before loading
- lfn/NR - no rewind before loading

EXECUTE. Completes loading, fills unsatisfied references by system (and user) library search, generates load map and executes program.

EXECUTE(pname,plist)

Begin execution at the specified entry point pname.
See LOADER, 2-4.

LGO. Load and execute the default compiler binary output file (See 'name.' below).

name.

name(plist) Load and execute binary program in lfn <name>. For a multiple load job, the last file loaded may use this format to initiate execution as well.

For example: LOAD(MYBIN)

LGO.

Parameters may be supplied to the called program. FTN5/FTN4 PL line limit may be reset (see page 13-4, 13-7, 13-30). File names defined in a FORTRAN program statement may be overridden in sequential order. For example:

LGO(,NEWOUT)

substitutes NEWOUT for the third file named. If <name> is not a binary program, an octal and character dump of the first ten words will appear in the dayfile along with the error message.

NOGO.

NOGO(lfn)

Completes loading of a program, including generating load map, but bypasses execution. <lfn> will contain the loaded program as a single core image module (non-segmented/ non-overlay loads only). If <lfn> is cataloged for production use, the absolute program will reduce to minimum memory in execution. <lfn> is suitable for inclusion in a user EDITLIB.

LIBLOAD(libname,eptnamel,eptname2,...,eptnamen)

Will load modules from specified library which contain the specified entry points. For core image load, only one entry point may be given. See LOADER, 2-3.

SLOAD(lfn,namel,name2,...,namen)

Will selectively load modules from file <lfn>. For core image load, only one entry point may be given.

<lfn> may have the following forms:

- lfn - rewind (except INPUT) before load
- lfn/R - rewind before load
- lfn/NR - no rewind before load

SATISFY(libnamel,libname2,...,libnamen)

SATISFY Satisfy unsatisfied externals prior to normal satisfaction at load completion; satisfy externals from user-specified libraries in the order specified. Cannot be used in segmented loads.

AD-A150 162

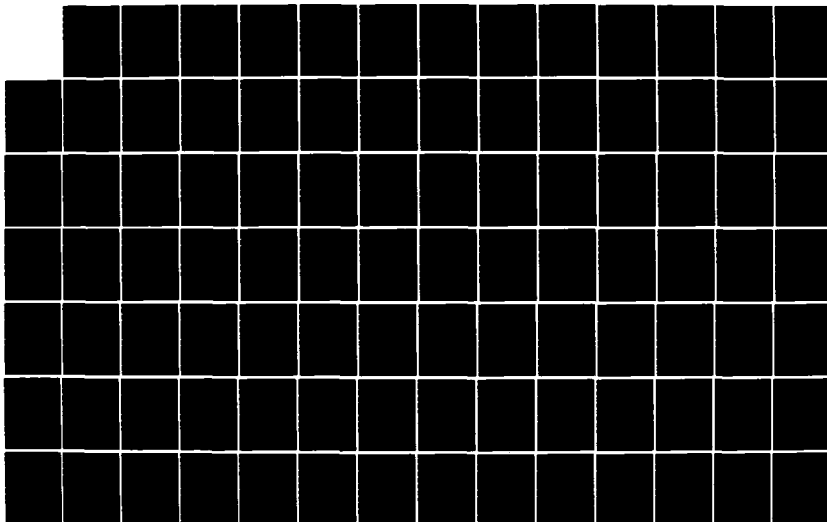
COMPUTER CENTER CDC REFERENCE MANUAL(U) DAVID W TAYLOR
NAVAL SHIP RESEARCH AND DEVELOPMENT CENTER BET..
D V SOMMER ET AL. SEP 84 DTNSRDC/CHLD-84-10

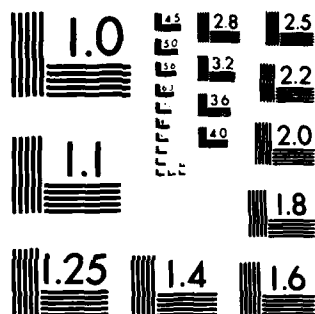
2/5

UNCLASSIFIED

F/G 9/2

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963 A

*** LDSET ***

LDSET(option1,option2,...,optionn)

Will set any of several options for the current load only.

Each option has one of the following forms:

key
key=param
key=param1/param2/.../paramn

Some of the options are:

FILES=lfnl/lfm2/.../lfnn

Permits Record Manager users to assure that system routines are loaded for the processing of specified files (page 3-30).

LIB=libname1/libname2/.../libnamen

Specifies one or more EDITLIB libraries comprising the local library set. (contrast with LIBRARY control statement (page 8-8).)

MAP=p/lfm Specified load map generation.

MAP=/lfm <lfm> is the file to contain the load map.

MAP=p (default is OUTPUT)

<p> is one or more of the following letters (default: SB):

N - no map

S - error messages and loader statistics

B - a block list and a list of unsatisfied externals

E - a list of entry points (without cross-reference map)

X - a list of entry points (with cross-reference map)

PRESET=p Preset memory in the specified manner.

PRESETA=p (PRESETA also puts each location's address in the 17 low order bits.)

Some values of <p> are:

P octal preset value

NONE no presetting

ZERO 0000 0000 0000 0000 0000

DEBUG 6000 0000 0004 0040 0000

NGINF 4000 0000 0000 0000 0000

ALTZERO 2525 2525 2525 2525 2525

(this is alternating 0 and 1 bits)

<p> may also be a 1-20 digit octal constant with optional + or - and optional B suffix.

(Default: PRESETA=DEBUG)

For additional values, see LOADER, 2-7.

USE=eptnam1/eptnam2

USEP=rtnam1/rtnam2

Used in overlay jobs to force loading of
certain Record Manager routines into
overlay 0,0 when loader error NE4340 occurs.

USEP=blkdat1/blkdat2

Used to force loading of BLOCK DATA
subprograms from EDITLIB user libraries.

For more options, see LOADER, 2-7 thru 2-9.

*** Loader-related Control Statements ***

MAP(p) Specify default option for load maps. It remains in effect until changed by another MAP statement. It may be overridden for the next load by using the 'MAP=' option of the LDSET statement. The more map requested, the more CP time and memory required to generate it.

<p> has one of the following values:

- OFF - no map (same as LDSET,MAP=N)
(Intercom default at DTNSRDC)
- PART - statistics, block map (same as LDSET,MAP=SB)
(batch default at DTNSRDC)
- ON - PART plus entry point cross-reference
(same as LDSET,MAP=SBX.)
- FULL - ON plus entry point map (same as LDSET,MAP=SBEX)

MAP. Reset to system default (batch: PART; Intercom: OFF)

MAP may occur within a load sequence.

LIBRARY(libname)

LIBRARY(libname1,libname2,...,libnamen)

LIBRARY. Specify a set of global libraries to be searched for externals and programs and the order in which the libraries are to be considered. This set remains in effect until end-of-job or until another LIBRARY statement is encountered.

The order of search for externals is:

- global libraries (those on most recent LIBRARY statement)
- local libraries (those in LIB= parameter of LDSET or LDSET tables in relocatable modules)
- system library ('SYSLIB')

The order of search for programs is:

- local files
- global libraries
- local libraries
- system library ('NUCLEUS')

The maximum number of user libraries which may be specified on a LIBRARY statement is 2. (With 2 user libraries, only 2 system libraries can be specified.) If more than 2 user libraries are required, then 'LDSET,LIB=...' must be used. See page 8-6.

'LIBRARY.' nullifies the effect of the previous LIBRARY statement. LIBRARY may not occur within a load sequence.

REDUCE. Turns reduce flag on. It remains on until end-of-job or until an RFL statement is encountered. The loader uses as much FL as needed for the load (up to job card CM or maximum Intercom FL), then just before program execution, reduces FL to minimum needed to contain the loaded modules.

type load	minimum FL
-----	-----
basic	Highest address of all programs and blocks (including blank common).
overlay	Highest address of all overlays.
segmented	For a program with blank common: Blank common follows the highest segment. Minimum FL is the end of blank common. For a program without blank common but with multiple levels: Minimum FL is end of highest segment in highest level. For a program without blank common and only one level: Minimum FL is end of root segment. FL is adjusted dynamically as segments are loaded and unloaded.

In all cases, FL is rounded up to a multiple of 100 octal words. REDUCE may occur within a load sequence.

RFL,xxxxxx. Request a new octal field length during the job execution (up to job card CM). RFL turns off the reduce flag for the next loading sequence. Therefore, a REDUCE statement must be used after the RFL to turn the flag back on, if desired.
RFL sets the maximum FL a program may use until another RFL is encountered.

Example:

```

REDUCE.      <-- turn on reduce flag
LG01.        <-- load, reduce FL to minimum, execute
RFL,60000.   <-- increase FL for next load, turn reduce
               flag off for next load
LG02.        <-- load and retain 60000B CM
ATTACH,LG03,ID=xxxx.
LG03.        <-- load in up to 60000 (last RFL),
               execute
RFL,70000.   <-- increase FL for next load, turn reduce
               flag off for next load
REDUCE.      <-- turn reduce flag back on
LG04.        <-- load in up to 70000 (last RFL),
               reduce FL, execute

```

RFL may not appear within a load sequence.

*** Segmentation ***

To implement segmentation a separate directive record is prepared to describe the tree structure. The modules will be loaded automatically as needed. Job field length is adjusted dynamically if program has no blank common, no level statements and is not in RFL mode.

All necessary Record Manager routines must be in root segment.

SEGLOAD, I=lfndir, B=lfnabs, LO=lfnout.

Initiate a segmentation job and read the directive record.

I=lfndir - directives are on file <lfndir>

(default: I=INPUT)

B=lfnabs - binary absolute module will be on file

<lfnabs> (default: B=ABS)

LO=lfnout - output list of tree and directives

(default: LO=OUTPUT)

Other LOAD and LDSET statements follow this statement.

** Segload Directives **

x TREE y

To define a tree structure.

<y> may be comma-separated list of other trees (pre-defined), segments or names of individual subprograms to be assigned a common starting address.

x TREE f-(c,d)

To indicate branching of the tree use -, then all following items are enclosed in parentheses.

c INCLUDE a,b

To assign programs <a> and to segment <c>.

Copies of a routine may be in different segments.

c GLOBAL com1,com2

To establish named commons at desired segment.

Reference name to left of directive must be defined by a previous directive.

c GLOBAL com1,com2-save

To save global block on disk for later calls to the segment which contains it.

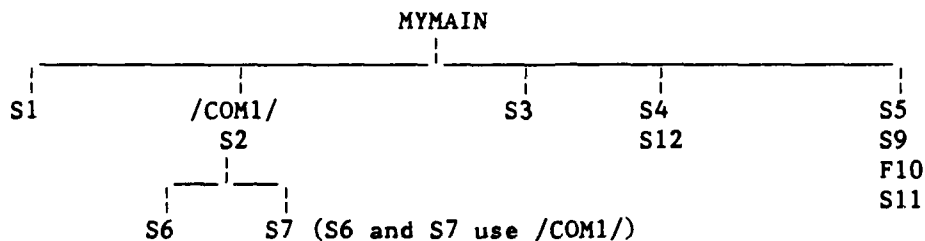
END ept

Should be the last directive in the record, where <ept> is the entry of the main program in the root segment. Non-fatal error if omitted.

See CLIB/P for public-access procedures to facilitate segmenting (SEGLD, SEGO).

**** Sample Tree Diagram ****

A block data subprogram defines common /COM1/ which is to be loaded with program S2. /COM1/ is also referred to by S6 and S7.



```

SEG INCLUDE S2,BLKDAT.
SEG GLOBAL COM1
PLUM TREE SEG-(S6,S7)
PEAR TREE MYMAIN-(S1,PLUM,S3,S4,S5)
S5 INCLUDE S9,F10,S11
S4 INCLUDE S12
END
  
```

By using nested parentheses one TREE directive may be eliminated.

*** Segmentation Cautions ***

1. To develop a segmented job, several runs may be required, so relocatable object code should be cataloged. Common blocks and Record Manager routines may need to be INCLUDED in lower segments to operate properly.

2. The load map must be checked carefully for any duplicate common block entries. Each common block which is referenced in more than one segment must be put into a global at the nearest-to-the-root segment. If any common block appears more than once without "safe", a global is required to eliminate duplicate storage areas. If input/output is performed in several segments, some Record Manager common blocks may be multiply defined (e.g., AOB.RM or Q8.IO.).

Subfields in SEGLOAD directives which contain any of the special characters , - () or which start with a \$ must be defined as literals (i.e., delimited by \$...\$). Embedded \$ is represented by \$\$, thus, \$TAN must be specified as \$\$TAN\$.

When Record Manager common is global to a root segment, the loader may detect errors in initializing. If so, an INCLUDE directive will be required to move the RM routines to that segment (e.g., 'MYMAIN INCLUDE INCOM=').

3. Directives must not go beyond column 72 of a card/line. They may be broken almost anywhere and continued on the next one or more cards/lines. The continuations have a comma (,) in column 1 as the continuation signal, then the directive is continued starting in column 2.

Continued directives should be avoided, if possible, to improve readability.

4. FTN5/FTN4 users should avoid passing external references as subprogram arguments. When the external is not in the root segment or the same segment as the call, execution will generate the fatal error message 'NON-EXECUTABLE WORD LOADING A SEGMENT'.

5. If the object module for a BLOCK DATA subprogram is located in an EDITLIB library, a LDSET statement with the USEP keyword is needed. For the example on page 8-11,

LDSET,USEP=\$BLKDAT.\$.

For multiple block data modules, use

LDSET,USEP=BLKA/BLKB/....

*** Overlay ***

(see LOADER, Chapter 6 or FTN5, Chapter 9 or FTN4, 7-19)

Overlay allows a user to fit a large program into a smaller amount of memory. Each overlay contains a main program (and optionally subprograms) and is a self-contained absolute program loaded and executed without linking during execution. Overlays may be primary or secondary, allowing only self or upward references. Data communication between overlays is only through blank or labelled common.

Overlay directives begin in column 7 or later. The main program in each level transfers control back to the caller when the END statement is executed. The main program for each level may be a short one, perhaps created to call other subprograms in the level, or a former subprogram may be modified. The loader requires at least 4000 (octal) locations for loading the (0,0) overlay.

** Overlay Directives **
(part of FTN5 or FTN4 source program)

OVERLAY (lfn,i,j) precedes main program of each overlay;
where <lfn> is overlay file name
<i> is primary level (0-77 octal)
<j> is secondary level (0-77 octal)

CALL OVERLAY (lfnloc,i,j) call into memory and execute;
where <lfnloc> may be left Display Code
of <lfn> or a variable containing the <lfn>
in left display code, and <i>,<j> are 1-63
decimal.
For example, CALL OVERLAY('PROG1',1,0)
(see page 7-18 for use in a program in a
library.)

PROGRAM namei No argument list on sublevel program.
All files must be identified on program
statement in overlay (0,0).

Note: Calling in a secondary overlay will not automatically load the appropriate primary overlay.

*** Setting Up Overlay ***

Original program

```
PROGRAM TESOS(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT,TAPE14)
COMMON /NAMED/ X, Y, Z(10,100)
C  TESOS calls two independent subroutines S Good 07/26/83
  READ (5, 9, END=10) X, Y
  ...
  CALL BAB
  ...
  CALL CT (F, L, Q)
  ...
10 STOP
9  FORMAT (2F12.6)
  END

SUBROUTINE BAB
COMMON /NAMED/ X, Y, Z(10,100)
  ...
  DO 7 K=1,5
    ...
    DO 6 I=1,10
      WRITE (14) (Z(I,J), J=1,100)
6    CONTINUE
7  CONTINUE
  REWIND 14
  RETURN
  END

SUBROUTINE CT (F, L, Q)
REAL P(50,100)
IF (L .NE. 1) THEN
  ...
  Q = P(L,L)
ELSE
  DO 5 I=1,50
    READ (14, END=10) (P(I,J), J=1,100)
5  CONTINUE
10 CONTINUE
  ...
  END IF
  RETURN
  END
```

*** Sample Overlay ***
 (see also page 7-18)

```

OVERLAY (TESOS, 0, 0) **
PROGRAM TESOS(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT,TAPE14)
C TESOS calls two independent subroutines S Good 07/25/83
C ** endpunching indicates changes in original program
COMMON /NAMED/ X, Y, Z(10,100)
COMMON /CCOM / F, L, Q **
READ (5, 9, END=10) X, y
...
CALL OVERLAY ('TESOS', 1, 0) **
...
CALL OVERLAY ('TESOS', 2, 0) **
...
10 STOP
9 FORMAT (2F12.6)
END

OVERLAY (TESOS, 1, 0) **
PROGRAM BACA **
C overlay by dummy main program
CALL BAB
END
SUBROUTINE BAB
COMMON /NAMED/ X, Y, Z(10,100)
...
DO 7 K=1,5
...
DO 6 I=1,10
WRITE (14) (Z(I,J), J=1,100)
6 CONTINUE
7 CONTINUE
REWIND 14
RETURN
END

OVERLAY (2, 0) **
PROGRAM CTCA **
C overlay by modify former subprogram
COMMON /CCOM / F, L, Q **
REAL P(50,100)
IF (L .NE. 1) THEN
...
Q = P(L,L)
ELSE
DO 5 I=1,50
READ (14, END=10) (P(I,J), J=1,100)
5 CONTINUE
10 CONTINUE
...
END IF **

END
```

*** Overlay Capsules ***

(see LOADER, Chapter 8 or FTN5, Chapter 9)

Overlay capsules allow a user to fit a large program into a smaller amount of memory. The program consists of a main overlay containing the main program and zero or more subprograms followed by one or more overlay capsules, each containing one or more subprograms. The first subprogram in each capsule must be a subroutine.

** Capsule Directives **

Capsule directives are part of the FTN5 source program and begin in column 7 or later.

OVERLAY (lfn, 0, 0, OV=n)

Precedes the main program.

lfn is the overlay file name

0,0 is the overlay level (must be 0,0)

n is the number of capsules in the program

OVCAP.

OVCAP(lfn1) Precedes each capsule. Each OVCAP statement is followed by the subprogram(s) to be included in the capsule. The name of the capsule is the name of the first subprogram (which must be a subroutine) following the statement.

If the second form is used, the absolute program is written into file lfn1 by the loader. Note that if 'NOGO,lfn2.' is used in loading, the absolute program is written into file lfn2.

CALL LOVCAP (name)

Load capsule <name> (<name> is a character variable or constant). Execution is not initiated. Capsules are loaded from the same file or library as the main overlay.

CALL XOVCAP (name)

CALL XOVCAP (name, user_parameters)

Execute capsule <name>. If necessary, it will be loaded first. <name> is the name of the first subroutine in the capsule. <user parameters> are the parameters of the first subroutine in the capsule.

CALL UOVCAP (name)

Unload capsule <name>. Any capsule may unload any other capsule. Care must be taken not to unload a capsule before it has finished execution.

Note: Capsules may call subprograms in the main overlay, but may not call subprograms in other capsules, except by loading and executing the capsule.

*** Setting Up Capsules ***

** Original Program **

```
PROGRAM APPLE (OUTPUT=128)
PRINT '(''In main program APPLE'')'
CALL PEACH (1.2, 3)
END
```

```
SUBROUTINE KIWI
PRINT *, 'In subroutine KIWI'
RETURN
END
```

```
SUBROUTINE PEACH (X, I)
PRINT *, 'In subroutine PEACH with X=', X, ', I=', I
CALL PEAR
RETURN
END
```

```
SUBROUTINE BERRY
PRINT *, 'In subroutine BERRY'
CALL KIWI
RETURN
END
```

```
SUBROUTINE PEAR
PRINT *, 'In subroutine PEAR'
CALL BERRY
CALL KIWI
RETURN
END
```

** Sample Overlay Capsule **

```
OVERLAY (FRUIT, 0, 0, OV=3)      <-- define main overlay with
                                <-- three capsules

PROGRAM APPLE (OUTPUT=128)
PRINT '(' 'In main program APPLE' ')'
CALL XOVCAP ('PEACH', 1.2, 3)    <-- load/execute with parameters
CALL UOVCAP ('PEACH')           <-- unload
CALL UOVCAP ('PEAR')            <-- all 3
CALL UOVCAP ('BERRY')           <-- capsules
END

SUBROUTINE KIWI
PRINT *, 'In subroutine KIWI'
RETURN
END

OVCAP.                          <-- define capsule PEACH
SUBROUTINE PEACH (X, I)
PRINT *, 'In subroutine PEACH with X=', X, ', I=', I
CALL LOVCAP ('PEAR')            <-- load capsule PEAR
CALL LOVCAP ('BERRY')           <-- load capsule BERRY
CALL XOVCAP ('PEAR')            <-- execute capsule PEAR
RETURN
END

OVCAP.                          <-- define capsule BERRY
SUBROUTINE BERRY
PRINT *, 'In subroutine BERRY'
CALL KIWI                      <-- call routine in main overlay
RETURN
END

OVCAP.                          <-- define capsule PEAR
SUBROUTINE PEAR
PRINT *, 'In subroutine PEAR'
CALL XOVCAP ('BERRY')           <-- execute capsule BERRY
CALL KIWI                      <-- call routine in main overlay
RETURN
END
```

*** Compile, Load and Catalog Absolute Program ***

** Simple Load **

```

jobname,....                name/code
CHARGE,....
FTN5,OPT=1.
REQUEST,MYPROG,*PF.          <-- MYPROG for absolute module
LOAD,LGO.
NOGO,MYPROG.                 <-- absolute module onto MYPROG
CATALOG,MYPROG,ID=xxxx,XR=RDONLY,PW=RDONLY.
<eor>
    PROGRAM MYPROG (...)
    ...
<eoi>

```

** SEGLOAD **

```

jobname,....                name/code
CHARGE,....
REQUEST,LGO,*PF.
FTN5,OPT=1.
CATALOG,LGO,MYSEGLGO,ID=xxxx. <-- save relocatable modules for
                                possible re-segmentation
REQUEST,MYSEGL,*PF.          <-- MYSEGL for absolute segments
SEGLOAD,B=MYSEGL.
LOAD,LGO.
NOGO.                        <-- absolute segments onto MYSEGL
CATALOG,MYSEGL,ID=xxxx,XR=RDONLY,PW=RDONLY.
<eor>
    (FTN5 source program)
<eor>
    (SEGLOAD control directives)
<eoi>

```

** Overlay Load **

```

jobname,....                name/code
CHARGE,....
FTN5,OPT=1.
REQUEST,MYOVLY,*PF.          <-- MYOVLY for absolute overlay
LOAD,LGO.
NOGO.                        <-- absolute overlay onto MYOVLY
CATALOG,MYOVLY,ID=xxxx,XR=RDONLY,PW=RDONLY.
<eor>
    OVERLAY (MYOVLY, 0, 0)
C    'MYOVLY' is the name of the absolute overlay file
    PROGRAM MYOVLY (...)
    ...
<eoi>

```

**** Overlay Capsule Load ****

```
jobname,....                               name/code
CHARGE,....
FTN5,OPT=1.
REQUEST,MYOVCP,*PF.                        <-- MYOVCP for absolute overlay
LOAD,LGO.                                  <-- absolute overlay onto MYOVCP
NOGO.
CATALOG,MYOVCP,ID=xxxx,XR=RDONLY,PW=RDONLY.
<eor>
OVERLAY (MYOVCP, 0, 0, OV=5)               <-- sample showing 5 capsules
C 'MYOVLY' is the name of the absolute overlay file
PROGRAM MYOVCP (...)
...
<eoi>
```

***** DISK FILES *****

*** Introduction ***

The CYBER system is a file-oriented system. A file may be actually a magnetic tape, a disk area, a disk pack, or any other such device.

Every sequential file is treated as though it were assigned to a magnetic tape (see chapter 11). Random access files must be on Rotating Mass Storage (RMS) devices.

Assignment of files to actual devices is usually accomplished outside the program with control statements. A file need not be assigned to the same device or even to the same type of device on different runs of the same program. If a file is not assigned to a particular device with control statements, it is automatically assigned to a disk area. It is recommended that for purely scratch files, disk files be used. Please place frequently used files on disk. Less-frequently used files and backup copies of files may be placed on the Mass Storage System (see chapter 10).

Information from an unblocked stranger tape to be stored on a permanent file or used as input to a compiler, must be copied to a system standard (SI) file. Card and print-line images as well as blocked stranger tapes may be converted using COPYRM (page 12-7) or catalogued procedure COPYBLK (CLIB/P) which utilizes form.

*** Record Manager ***

Record Manager (RM) is a unified I/O package used by major product sets. COBOL and Compass may define many file types within the language. FORTRAN utilizes such files by means of FILE definition statements.

RM supports five (5) types of file organizations: sequential (SQ), indexed sequential (IS), word addressable (WA), direct address (DA), and actual key (AK). (See RMFTN, RMUG)

All RM routines should be in the 0,0 overlay of overlay jobs.

*** File Organizer and Record Manager ***
(FORM)

The FORM utility can manipulate records and reorganize files. Records may be reblocked from stranger files. Test data may be generated by selecting records from a larger input file. All I/O is handled by Record Manager. The user must supply proper FILE statements and directives. See FORM Reference Manual and CCRM, 12-7: COPYRM.

*** Characteristics of Rotating Mass Storage Devices ***

On-line rotating mass storage on the DTNSRDC CDC CYBER computers is single and double density 844 disks, 885 disks and 819 disks. On 844 drives, positioning may take place on one drive while reading or writing another (seek overlap). Permanent files reside on model 885 disks. Some of the single density removable packs are available as user device sets (see page 5-4: RP parameter). The NOS/BE operating system, queues and scratch files reside on 844 packs. These packs have single error recovery and are very reliable. On the CYBER 176, scratch files reside on model 819 disks.

It is the user's responsibility to define the file structure desired. The operating system allocates space as needed in units of record blocks (RB). Data is written into physical record units (PRU) when a file is generated, until all the PRU's in the RB are filled, then another RB is allocated. Although PF storage charges are for PRU's, actual space is allocated in whole RB's. Since more time is spent in positioning than transmitting data, a memory request stack contains all outstanding mass storage file action requests and the least positioning operation is performed first. Each single density removable disk pack holds over 115 million characters. Some of the system packs are double density and hold twice as much. Each single density 844 disk contains 3232 RB's with 57 PRU's per RB. Each 885 disk contains 6712 RB's with 160 PRU's per RB for a total of over 692 million characters.

FORTRAN and COBOL process all read/write operations through buffers. The programs manipulate user-defined records such as coded unit records or logical binary records. The NOS/BE system manipulates disk PRU's of 64 words. Trailing blanks are stripped from coded records and a line terminator (12 bits of binary zero) is added before a record is placed in a buffer for unit record devices.

Several random access techniques are available for data files which are not sequential in nature. Note that accessing many small pieces of data is time consuming and it may be better to read sequentially, since both COBOL and FORTRAN buffer I/O. Job throughput may be decreased by poor choice of record and block structure.

*** Permanent Files ***

Permanent files are data or program files permanently stored on selected disk devices and accessed by the NOS/BE commands given below. It is the originator's responsibility to purge a file when it is no longer needed.

There are two separate sets of permanent files: one is on the CYBER 176 and one is on the CYBER 750. To transfer files between CDC mainframes, use the Mass Storage System (chapter 10).

Only unclassified programs (source or object) and data may be cataloged into the public permanent file system. New files will be dumped to backup tape daily and held one week. The total permanent file base is dumped once a week and retained about 3 weeks.

Occasionally some permanent files become unreadable and not all files may be recoverable. The user should have provision for recreating the files if this situation occurs. (See CLIB/P: SELDUMP, SELLOAD)

Permanent file commands may be continued onto additional lines/cards by filling all but the last line/card through column 80 (or stopping after a comma) and starting the next line/card in column 1.

*** Permanent File Hints ***

1. The CDC computers run slower when space for user files is in short supply. The system will stop entirely when too much space on packs is occupied by permanent files and large scratch files.

The following should be done to help reduce the number of permanent file names in the system:

- a. Use EDITLIB to put subprograms into one library, absolute programs into another, CCL procedures into a third (such as the system libraries NSRDC, UTILITY and PROCFIL).
- b. Use UPDATE to maintain source programs in one or a few libraries.

2. Any job which attaches or catalogs many files should be run at P1 (overnight) to reduce PF directory contention during the prime shift. Anyone who attaches all files monthly to avoid the Computer Center automatic purge should not, since this process only accesses the PF directory and does not verify the contents of the files.

*** Automatic Permanent File Purge ***

Permanent files may be purged by the Computer Center for one of the following reasons:

- 1) invalid or cancelled job order number for AC
- 2) invalid ID (e.g., not user initials)
- 3) not accessed within the last 30 days
- 4) exceeds the maximum size of 5000 PRUs

The non-accessed files which are purged will be kept on tape for at least one (1) month. Anyone wishing to recover such a purged file should run procedure PFRSTOR (see CLIB/P) and then call the Permanent File Librarian, Code 1892.1, (202) 227-1907. A nominal fee will be charged for this service. After the files have been restored, you must use them to prevent another purge.

Note that an unneeded file should be purged by the user rather than waiting over 30 days for the automatic PURGE, since there is a charge for permanent file storage as long as the file exists as a permanent file in the system.

The expiration date is not presently used for file purging. However, the user may use this parameter as a reminder of the intended life expectancy of a file. The default expiration date is thirty (30) days after creation. Infinite retention is reserved for system files.

See pages 9-6: RENAME and 6-6: RENAMAC to change the job order number on your files.

*** User Audit ***

An audit program is provided to determine what files a user has on permanent file storage. The output list is of the permanent files existing at the moment the AUDIT program is run. (see page 9-15, for the user device set audit.)

An audit is obtained by one of:

unsorted AUDIT,<keyword-parameters>.
sorted BEGIN,AUDIT,,<keyword-parameters>.

AUDIT. implies AUDIT,ID=<id-from-LOGIN>,AI=I. Intercom
 AUDIT,ID=<id-from-CHARGE-card>,AI=P. batch

The following parameters, all optional, are available:

AC= Audit by job order number.
AC=jjjjjjjjjj - full 10 digits
AC=jjjjjjjj00 - first 8 digits (all segments audited)
AC - job order number corresponding to CHARGE card
 or LOGIN access number
 (for sorted audit use: AC=AC)

AI= Type of audit desired (ID or PF/ID only).
AI=F -- Full Audit (two lines per file)
 (prus, id, cy, full pfn, ac, creation date, date/time
 last attached, date/time last altered, # uses, # rbs,
 # alters/rewrites, # extends, expiration date, disk_vsn)
AI=I -- Intercom Audit (one 72-column line per file)
 (id, prus, cy, 33-characters of pfn, ac, creation date,
 date last attached, # uses) (Intercom default)
AI=P -- Partial Audit (one line per file)
 (prus, id, cy, full pfn, ac, creation date, date last
 attached, date last altered, # uses) (Batch default)
AI=S -- Short Audit (id, cy, full pfn)

ID= Audit files of given user(s). Up to 10 ID's (7 for sorted) may
 be specified. For example,
 unsorted: ID=<id1>/<id2>/.../<id10>
 sorted: ID=\$<id1>/<id2>/.../<id7>\$,IDM=YES
 (If ID (and AC) omitted, ID is taken from CHARGE card or LOGIN)

LF= Output file (default: OUTPUT)

PF= Single permanent file name to be audited (all cycles)
 (default: all pfn's are audited)
 (If AC and PF are specified, PF is ignored.)
 (If PF without ID is specified, ID is taken from the CHARGE
 card/LOGIN.)

In addition, several sorting options are available.

*** Permanent File Commands ***

REQUEST, lfn, *PF.

Must precede the initial writing of file lfn.

CATALOG, lfn, pfn, ID=xxxx, AC=jjjjjjjjjj, <parameters>.

CATALOG, lfn, ID=xxxx, AC=jjjjjjjjjj, <parameters>.

Enter the file in permanent file directory after writing.

ATTACH, lfn, pfn, ID=xxxx, <parameters>.

ATTACH, pfn, ID=xxxx, <parameters>.

To use a previously cataloged file.

PURGE, lfn, pfn, ID=xxxx, <parameters>.

PURGE, pfn, ID=xxxx, <parameters>.

If the file to be purged has been attached with control permission previously in the job, only <lfn> is required.

CAUTION

If <lfn> is already a local file, the rest of the statement is ignored and the purge is attempted on the existing <lfn>.

RENAME, lfn, pfn, <parameters>.

Changes name or alters passwords, cycle, or AC. However, file must be attached with all permissions before RENAME. Cannot change <pfn>, <id> or <pw> for one cycle of a multi-cycle file. Cannot rename if MR=1. To rename <id>, use procedure NEWID or RENAMID (see CLIB/P).

EXTEND, lfn.

Add permanent data to the end of an attached permanent file.

ALTER, lfn.

Shorten or lengthen an attached permanent file. End-of-information will be reset to the current position in the file. Once a file is shortened, the deleted information cannot be retrieved!

** Required Parameters **

lfn Local file name, 1 to 7 characters, first must be alphabetic.

pfn Permanent file name for system CATALOG, up to 40 characters.
If lfn is not specified, first 7 characters of pfn are lfn.
If pfn is not specified, lfn is also pfn.

ID=xxxx Registered initials of file owner (as on CHARGE card).
Required (for non-public-access files) on CATALOG, ATTACH, and PURGE (long form).

**** Random-file Parameter ****

FO= Use on CATALOG of advanced access method (AAM) files to eliminate possible updating problems.
FO=AK (actual key)
FO=DA (direct access)
FO=IS (indexed sequential)
Note that FORTRAN READMS/WRITMS does not use this parameter.

**** Other Parameters (usually optional) ****

All optional permanent file parameters are keywords. A keyword is followed by '=', then the value with no blanks.

AC=jjjjjjjjjj Job order number to charge storage of file. When omitted or invalid format (e.g., too few characters), AC corresponds to the CHARGE-card/LOGIN access number.

CY= Cycle number 1 to 999 (initial create default is 1).
Only five (5) cycles may exist at any one time.
Note than the highest numbered cycle will be attached or purged if CY and LC are not specified.

LC=1 Access the lowest numbered cycle on ATTACH or PURGE.

MR=1 Allows multi-read access on ATTACH (simultaneous use by different jobs). Also, job may be rerun by system if error. A file attached with read-only permission cannot be PURGED, RENAMED or EXTENDED.

RP= Retention period in days, up to 720. (default: 30)

TK= Turnkey password (controls access to the file).

RD= Read password.

XR= Password definition for read-only files. Sets CN, MD, EX.
Used on CATALOG instead of MR=1 on ATTACH.

EX= Extend password.

MD= Modify password.

CN= Control password.

Note: The above passwords, which apply to all cycles of a file, are 1-9 alphameric characters and are used only to define passwords on CATALOG or RENAME.

PW=list Used to submit passwords. A maximum of five (5) passwords for ATTACH or for CATALOG of a new cycle.

*** Hints on Permanent File Usage ***

1. Attempt to purge a non-existing file will be flagged but non-fatal.
2. CATALOG without CY= parameter for pre-existing file will catalog the next higher cycle if control permission is granted.
3. Files should be cataloged with XR and PW passwords to take full advantage of multiple read on ATTACH and multiple cycles on CATALOG.
4. System permanent files will have public access ID and may be attached by pfn alone.

** Examples **

CATALOG,LGO,SPSOBJ,ID=xxxx,AC=1111111111,XR=READ,PW=READ.
(identical CATALOG will create an additional cycle)
ATTACH,SPSOBJ,ID=xxxx. <-- only read permission, lfn is pfn
PURGE,OLD,SPSOBJ,ID=xxxx,PW=READ,LC=1. <-- discard lowest cy
ATTACH,NSRDC5. <-- a system library, ID not required
CATALOG,ANYCPL,ID=xxxx,XR=RO,PW=RO. <-- job AC used

*** Permanent File User Return Codes ***

Permanent file errors are referenced by the error number. Intercom references are decimal codes.

decimal	octal	meaning	related commands
0	000	Function successful.	
1	001	PFN/ID error.	
2	002	Lfn already in use.	ATTACH
3	003	Unknown lfn.	CATALOG, EXTEND, PURGE, RENAME
4	004	No room for extra cycle (5 max).	CATALOG
5	005	PF catalog full.	CATALOG, EXTEND
6	006	No lfn or pfn.	
7	007	(not used)	
8	010	Latest index not written for a random file.	CATALOG, EXTEND
9	011	File not on a PF device.	CATALOG
10	012	File not cataloged, SN=<setname>.	ATTACH
11	013	Archive retrieval aborted.	ATTACH
12	014	Bad LPF communication.	CATALOG, RENAME
13	015	CY # limit reached (999).	CATALOG
14	016	PF directory full.	CATALOG
15	017	Function attempted on non-permanent file.	CATALOG, EXTEND, PURGE, RENAME
16	020	Function attempted on non-local file.	
17	021	Improper archive retrieval call.	ATTACH
18	022	File never assigned to a device (file never written on).	CATALOG
19	023	Cycle incomplete or dumped.	ATTACH
20	024	File already attached.	ATTACH
21	025	File archived.	ATTACH
22	026	Illegal character in FDB parameter.	
23	027	Illegal lfn.	
24	030	File dumped.	ATTACH
25	031	Illegal function code.	
26	032	Purge attempt ignored; use RB param.	PURGE
27	033	ALTER needs exclusive access.	ALTER
28	034	FDB is too large.	
29	035	File already in system.	CATALOG
30	036	No APF space.	ATTACH
31	037	Permission conflicts.	
32	040	Illegal setname specified.	
33	041	Device set not mounted at control point.	
34	042	RBT chain too large for PFC.	
35	043	File resides on unavailable device.	ATTACH, PURGE
36	044	File not available.	ATTACH, PURGE
37-55	045-067	(not used)	
56	070	PFM stopped by system.	
57	071	Incorrect permission.	
58	072	FDB address invalid.	
59	073	I/O error on PFD/PFC read/write.	

*** Permanent File Examples ***

** Construct Permanent File From Cards **

```
jobname,....                name / code
CHARGE,....
REQUEST,PROG,*PF.
COPYCR (INPUT,PROG)          <-- or COPYR,INPUT,PROG,C.
CATALOG(PROG,PROGRAM,ID=xxxx,CN=reserve) <-- prevent unintentional purge
<eor>
    (all decks to be made part of permanent file)
<eoi>
```

** Use of Permanent Source File **

```
jobname,....                name / code
CHARGE,....
ATTACH,USEIT,PROGRAM,ID=xxxx,MR=1.    <-- allows multiple read
FTN5(I=USEIT,OPT=1)
    .      other control statements as required
    .
    .
<eor>
    (data cards, if required)
<eoi>
```

** Purge Obsolete File **

```
jobname,....                name / code
CHARGE,....
PURGE,NOGOOD,OLDFILE,ID=xxxx,PW=reserve) <-- allow control permission
<eoi>
```

** Catalog New Cycle of Existing File **

```

jobname,....                      name / code
CHARGE,....
REQUEST,NEWCYC,*PF.
ATTACH,PROG,PROGRAM,ID=xxxx,PW=RESERVE.
                                <-- no multiple read until changed
COPYR(INPUT,NEWCYC)
COPYF(PROG,NEWCYC)              <-- add old info to newcyc
CATALOG,NEWCYC,PROGRAM,ID=xxxx,PW=RESERVE.
                                <-- will assign next cycle number
                                (up to 999)
<eor>
    (new cards to be put into next cycle)
<eoi>

```

** Replace Job Number, Add Permissions **

```

jobname,....                      name / code
CHARGE,....
ATTACH,PROG,PROGRAM,ID=xxxx,PW=RESERVE.      <-- control permission
COMMENT.    Note: multiple read must not be allowed.
RENAME,PROG,AC=newjobnbr,EX=RESERVE,MD=RESERVE.
COMMENT.    pf name stays the same - all cycles
COMMENT.    new job order number - this cycle
COMMENT.    additional passwords - all cycles
<eoi>

```

** Extend Existing File **

Information may be added to an existing permanent file only at the end-of-information (eoi). This example illustrates adding one (system) record to a file. The SKIPF will bypass the (nnn) existing records. If an end-of-file exists, the COPYR will abort since the file is not at EOI, so the alter is needed to reposition the EOI (any information past this point in the original file is lost). Note that ALTER will put three lines beginning with EX into the dayfile.

. other control statements as required

```

ATTACH,DATA,ID=xxxx,PW=LARGE.      <-- extend permission
COMMENT.    Note: multiple read must not be allowed.
SKIPF,DATA,nnn.                    <-- position to end of data
COMMENT.    (past nnn (system) records)
ALTER,DATA.                         <-- reposition eoi
COPYR,INPUT,DATA.                  <-- add new record
EXTEND,DATA.                        <-- reposition eoi
<eor>
    (new data record to be added)
<eoi>

```

**** Shorten Existing File ****

If a file contains valid information in the first part (one or more system logical records) and unwanted information at the end, the extra data may be removed, that is, the file shortened, by using the ALTER command. This will reposition the end-of-information (eoi) to the current position in the file, permanently removing any trailing information.

. other control statements as required
.

ATTACH,DATA,ID=xxxx,PW=LARGE. <-- extend permission
COMMENT. Note: multiple read must not be allowed.
SKIPF,DATA,nnn. <-- position to end of data to be kept
ALTER,DATA. <-- reposition eoi

*** Device Sets ***

On DTNSRDC CDC computers, some of the removable disk packs are available for users (page 9-2). The owner of a device set (DS) has all capabilities necessary to manage his own permanent file system. Each pack may contain up to 175,000 PRUs depending on the number of flaws and the size of the directory. User files may be written to a DS and made permanent using standard permanent file control statements with one additional parameter. A DS may be used by several jobs simultaneously. The standard AUDIT commands containing one additional parameter is used to list the contents of the pack. All AUDIT options are available. Each pack has a unique volume serial number (VSN).

Device sets cannot be used for on-line storage (use permanent files instead). NOS/BE treats device sets the same as magnetic tapes in that there are a limited number of drives for device sets and jobs requiring them must wait until the required drive(s) are available.

It is the user's responsibility to provide backup (tape or disk) for device sets. The Computer Center provides periodic backup for files on public devices only.

Device set packs are assigned by the Tape Librarian, Code 1896, (202) 227-1227. With information provided by the owner, User Services must initialize the pack with a label, write system flaw messages so that data will not cause uncorrectable RMS errors, and create on the DS a special permanent file DUM to allow backup dump capability. Information which must be provided includes:

- master pack (MP) VSN, usually same as VOLSER
- VOLSER the unique six character VSN
- set name (SN) 1-7 alphanumeric starting with a letter
- retention period for the pack (RP) up to 720 days
- unique RD, MD, CN, and EX passwords for the DUM file as required for mode 1 tape backup dumps, TRANSPF or LOADPF
- job order number (AC) for catalog of DUM

The default number of files in a DS directory is 400. If you need more or significantly fewer, an NF parameter is also required. Beginning at level 508, the owner may select unique universal (UV) and public-access (PB) passwords for each pack. (See NOSBE, Chapter 4)

*** Using Older Device Sets ***

Any device set initialized prior to level 508 must have procedure ADDEXT applied to it before DUMPF, LOADPF or TRANSPF may be used. (See CLIB/P: ADDEXT.)

*** Device Set Control Statements ***

PAUSE. JOB REQUIRES DISK PACK volser.

Notify operator to prepare required pack for mounting.

MOUNT,VSN=volser,SN=setname.

MOUNT,VSN=volser,SN=setname,PW=userpass.

Request an existing DS to be physically mounted (if not already mounted) and logically associated with this job. The MOUNT command must precede all other DS control statements except RECOVER (see pages 9-14, 7-6).

DSMOUNT,VSN=volser,SN=setname.

Logically disassociate the DS from this job. (Automatic at end-of-job.) Returns all files attached from the pack.

SETNAME,setname.

Establish a default SN for following ATTACH and REQUEST statements. Use only when all files to be attached reside on the pack.

SETNAME.

Re-establishes the system device as default for ATTACH and REQUEST.

REQUEST,lfn,*PF,SN=setname.

REQUEST,lfn,*PF,SN. (if preceded by SETNAME statement)

Request file <lfn> be written on the DS.

ATTACH,lfn,ID=xxxx,SN=setname,....

ATTACH,lfn,ID=xxxx. (if preceded by SETNAME statement)

Attach the indicated permanent file from the DS.

CATALOG statements and other ATTACH parameters are unchanged from system permanent file devices. Files are removed from the pack by PURGE.

*** Device Set Utilities ***

AUDIT,AI=F,SN=setname.

Audit all files on DS <setname>.

See page 9-5 for other AUDIT parameters.

DUMPF(PW=rdpassw,SN=setname)

Create backup tape dump of all files on DS.

See example for correct sequence of instructions.

LOADPF(SN=setname,PW=expassw)

Reload all files from tape to set.

LOADPF(I,SN=setname,PW=expassw)

Allows for selective load of files, using directives of the form: PF=pfnl,ID=xxxx.

RECOVER,VSN=volser,SN=setname.

If a machine failure occurs while a DS is mounted, this utility (not preceded by a MOUNT) validates the DS, verifies and recreates all critical disk tables. (call User Services)

TRANSPF

Utility to backup one device set onto another pack. See CLIB/P: TRANPK. (Must be done on block time since two packs are used simultaneously.)

** Backup Dump **

jobname,GE1,RP1,T500.

name / code

CHARGE,....

PAUSE. JOB REQUIRES DISK PACK DV9999.

MOUNT,VSN=DV9999,SN=setname.

VSN,DUMTAPE=CA8888/CA8889.

<-- DUMTAPE and L= must be as shown

LABEL(DUMTAPE,L=\$DUMPF TAPE- NEWS,F=S,W,D=GE,RING)

DUMPF(PW=rdpassw,SN=setname)

<eoi>

** Selective Reload **

... <-- first five lines of above example

LABEL(DUMTAPE,L=\$DUMPF TAPE- NEWS,F=S,R,D=GE,NORING)

LOADPF(I,SN=setname,PW=expassw)

<eor>

PF=myfile1,ID=xxxx.

PF=myfile2,ID=xxxx,CY=2.

<eoi>

*** Indexed Sequential ***

CYBER Record Manager with file organization type known as indexed sequential (FO=IS) replaced former SCOPE indexed sequential (SIS) version 2.0.

When cataloging an indexed sequential file, use FO=IS on the CATALOG statement to eliminate possible updating problems.

Example-- The following illustrates execution of COBOL program which creates a SIS file with the local file name of MYFIL and utilizes some of the above described options.

```
jobname,....          name / code
CHARGE,....
COBOL(LR,U,DB,Z)
REQUEST,MYFIL,*PF.
LGO.
CATALOG,MYFIL,MYSISFILE,ID=xxxx,XR=RDONLY,FO=IS.
<eor>
    (COBOL source program)
<eor>
    (data cards, if any)
<eoi>
```

The indexed sequential file utilities DUMPS, DUMPC, CREATE, DUMPR, and RELOADR are not included in NOS/BE or SCOPE 3.4 because similar capabilities are offered by Record Manager and FORM, and by NOS/BE utilities. The utilities ESTMATE and SISTAT are available.

In order to calculate the best block length for a SIS file, program CALCIBL and subroutine IBL are provided in libraries UTILITY and NSRDC, respectively. To print descriptive documents, use

```
BEGIN,DOCGET,,UTILITY,,CALCIBL,OUTPUT,MSACCES=<pw>.
BEGIN,DOCGET,,NSRDC,,IBL,OUTPUT,MSACCES=<pw>.
```

If a SIS file is to be saved for backup and the structure of the file is to be unchanged when reloaded, the COPYBF file utility may be used to dump and reload the file.

The following examples illustrate the use of FORM to perform SIS utility functions. (see FORM, 4-4, 5-14)

In this example the sequential records of a SIS file are copied to a sequential file.

```
jobname,.....          name / code
CHARGE,.....
FILE(SISFILE,FO=IS,KL=10,KT=I,MNR=1000,MRL=1000)
FILE(SEQFILE,FO=SQ,BFS=260,LT=UL,BT=C,RT=Z,MRL=1000)
LDSET(FILES=SISFILE/SEQFILE)
FORM.
<eor>
INP(SISFILE)
OUT(SEQFILE)
XEQ
<eoi>
```

In this example a SIS file is created from a sequential input file. The OUT directive in the FORM control cards indicates the key to be used in building the SIS file. In this case it will be a 10-character integer starting with character 1 of the input record.

```
jobname,.....          name / code
CHARGE,.....
FILE(SEQFILE,BFS=260,LT=UL,BT=C,RT=Z,MRL=1000,FO=SQ)
FILE(SEQFILE,KT=I,KL=10)
FILE(SISFILE,FO=IS,KL=10,MNR=1000,MRL=1000,KT=I,ERL=10)
FILE(SISFILE,DP=20,IP=20)
LDSET(FILES=SISFILE/SEQFILE)
FORM.
<eor>
INP(SEQFILE)
OUT(SISFILE,KEY=+1I10)
XEQ
<eoi>
```


***** MASS STORAGE SYSTEM (MSS) FILES *****

*** Mass Storage System (MSS) ***

The CDC Mass Storage System (MSS) is a large capacity on-line mass storage device. It is a cost effective extension to the NOS/BE disk file system and is an alternative to private packs (device sets) and conventional magnetic tape storage. Specifically, the MSS offers:

- . Approximately 12 times the on-line storage capability of the CYBER 750 and CYBER 176 systems.
- . On-line access (via Intercom) to files which previously had to be stored on magnetic tape (or private pack) because of size restrictions and/or infrequent use.
- . Reduced storage charges for these on-line files. Note that data is stored in one or more "streams" of 640,000 characters (1000 NOS/BE PRUs).

The following pages list the MSS commands. In addition, there are several procedures and programs which support the MSS.

For further information, see "Computer Center Mass Storage System User's Guide" and CLIB/P.

*** MSS Security ***

To provide adequate security for MSS users, you must submit your MSS password in any job or Intercom session which will manipulate MSS files. The initial setting for all users is the 4-character User ID. To protect your MSS files, you should change this initial password to one of your own choosing (4 to 7 alphameric characters) (see page 10-4: MSPASSW).

*** MSS File Purge ***

MSS files may be purged by the Computer Center if the job order number for AC is invalid or has been cancelled.

To recover any purged file, call the File Librarian, Code 1892.1, (202) 227-1907. A nominal fee will be charged for this service. After the files have been restored, you must rename the AC to a valid job order number.

See pages 10-4: MSCHANG and 6-6: RENAMAC to change the job order number.

*** MSS Backup for Critical Files ***

In addition to normal permanent or MSS file backup, critical files may be backed up and stored off-station. These files are available in the event of a catastrophe (such as fire) at the Computer Center in Carderock.

For a file to be designated as "critical", it must have the attribute CRIT set. This is done in the MSSTORE and MSCHANG statements by specifying "CRIT=Y" if the file is critical, or "CRIT=N" if it is not. The default is CRIT=N. For example:

```
MSSTORE,lfn,mfn,CRIT=Y.  <-- store a critical file
MSCHANG,mfn,CRIT=Y.     <-- make a file critical
MSCHANG,mfn,CRIT=N.     <-- make a file non-critical
```

Files designated for this off-station backup service will be charged a higher rate.

*** MSS Commands ***

MSACCES,<access-password>.

Required to gain access to the MSS. Access is granted until end-of-job or end-of-session. <access-password> is 4-7 alphanumeric characters, first alphabetic, and may be changed by MSPASSW.

MSAUDIT,LO=<lo>,PF=<mfn>,UN=<un>,LF=<lfn>,PW.

Audit a user's MSS files.

- LO=0 (zero) - alphabetical list of a user's MSS files.
- LO=F - LO=0 plus selected attributes.
- LO=FP - a summary of file accesses by other users (PF= is required).
- LO=P - like LO=FP, except only names of users who have access to <mfn>.

LF=<lfn> - local file to hold the MSAUDIT output.

If PW is specified, each file's password, if any, will be listed. (Only for LO=F and UN omitted or UN=<yourself>.)

(defaults: LO=0, LF=OUTPUT)

For a sorted LO=F audit, use:

BEGIN,MSAUDIT,MSS,LF=<lfn>,UN=<un>.

MSSTORE,<lfn>,<mfn>,CT=<ct>,PW=<pw>,AC=<ac>,CRIT=<crit>.

MSSTORE,<lfn>,CT=<ct>,PW=<pw>,AC=<ac>,CRIT=<crit>.

MSSTORE,<lfn>,<mfn>,CT=<ct>,PW=<pw>,AC=<ac>,CRIT=<crit>,NA=1.

MSSTORE,<lfn>,CT=<ct>,PW=<pw>,AC=<ac>,CRIT=<crit>,NA=1.

Store local file <lfn> on the MSS with permanent file name <mfn>. If <mfn> is omitted, then <mfn>=<lfn>.

If NA=0 or is omitted, the file will not be stored if <mfn> already exists as an MSS file. If NA=1, any existing MSS file <mfn> will be replaced.

(Defaults: CT=P, PW=0, AC=<CHARGE/LOGIN>, NA=0, CRIT=N)

MSFETCH,<lfn>,<mfn>.
MSFETCH,<lfn>.

MSFETCH,<lfn>,<mfn>,UN=<un>,PW=<pw>.
MSFETCH,<lfn>,UN=<un>,PW=<pw>.

Make a copy of MSS file <mfn> as local file <lfn>. If
<lfn> is omitted, <lfn>=<mfn>.

MSPURGE,<mfn>.
Remove MSS file <mfn>.

MSCHANG,<newmfn>=<oldmfn>,CT=<ct>,PW=<pw>,AC=<ac>,CRIT=<crit>.
MSCHANG,<mfn>,CT=<ct>,PW=<pw>,AC=<ac>,CRIT=<crit>.

Change MSS file attributes. Use the second form if the
permanent file name is not to be changed. To remove the
password, use PW=0.

MSPERMT,<mfn>,UN=<un>,M=<m>.

Define/change permissions explicitly for user <un>:

- 1) grant access (M=R (read only) or M=W (read, write,
purge)) to a private (CT=P) file to another user.
1a) deny access (M=N) to a user previously granted
access.
- 2) restrict access (M=N) to a semi-private (CT=S) file
by another user.

(Default: M=R)

MSPASSW,<old>,<new>.

Change MSS access password. <old> is the current access
password; <new> is the new one. Each is 4-7 alphanumeric
characters, first alphabetic.

** MSS Command Parameters **

<lfn> NOS/BE local file name, 1-7 alphanumeric characters, first alphabetic.

<mf> MSS permanent file name, 1-7 alphanumeric characters.

AC=<ac> 10-digit job order number to charge storage of the file. If omitted, it is taken from the CHARGE card/LOGIN.

CRIT=<crit> Critical file:
CRIT=N - The file is non critical and will have only local backup.
CRIT=Y - The file is critical and will also have off-station backup at a higher cost.
The default is CRIT=N.

CT=<ct> Category Type:
CT=P - Private. Only the owner and those explicitly granted permission have access.
CT=PU - Public. All users have access.
CT=S - Semi-private. All users, except those explicitly denied permission, have access. In addition, all accesses by other users are logged.

PW=<pw> Password, 1-7 alphanumeric characters. Required only if referring to another user's file.

UN=<un> User initials. Required only if referring to another user's file or for MSPERMT.
(Synonym: ID=)

***** TAPE FILES *****

*** Tape Characteristics ***

Magnetic tapes should be used for sequential data for such purposes as:

1. Transfer of information to and from other computers and off-line peripherals.
2. Files which are used infrequently.
3. Back-up copies of disk files.
4. Long-term storage of data.

Tapes should not be used for scratch files or random information. For safety, two copies on different tapes should be maintained, or for data which is updated, a grandfather-father-son system is advised. It is not wise to mount a tape containing good data, read through it, and write new data at the end. Instead, copy the existing data to a second tape and add the new data to the second tape, retaining the first tape as a back-up.

Processing a file on tape will take considerably more I/O time than on disk and more elapsed time.

Information concerning the physical and logical characteristics of the tape is specified on control statements:

1. The job card must indicate if any tapes are to be used during the job.
2. VSN statements identify the tape reels to be used.
3. LABEL or REQUEST statements give additional specifications, such as density and read-write status.

The LABEL or REQUEST statement should be placed just before the load of the program which requires the tape. Thus, if an earlier step fails, the job will abort without causing the operator to mount the tape.

** SI Tapes **

SI tapes are the system standard and should be used unless there are special reasons not to do so. In addition to any other structure, they are organized into system-logical-records. (A system-logical-record is most familiar as a sequence of unit records between two EOR's.) SI tapes have two standard physical record (PRU) sizes: 128 words for coded information and 512 words for binary. At the end of a system-logical-record there may be a shorter physical record. Physical record size is not ordinarily a concern of the user on SI tapes, as the system automatically assembles information into the standard record sizes.

** Stranger Tapes **

Stranger tapes differ from SI tapes in that stranger tapes are not organized into system-logical-records and in the available physical record sizes. They are most often used when going to or from non-CDC equipment. They may be labelled or unlabelled. Use of stranger tapes is specified by an S or L parameter on the REQUEST statement or 'F=S', 'F=L' in the LABEL statement. Stranger (S) tapes may have physical records of any desired length up to 512 words. L-tapes are a special case of stranger tapes which may have physical records longer than 512 words. Stranger tapes of small records, such as unblocked card images or print lines, take a disproportionately long time to read or write. If such a tape may be used more than once, it should be reblocked into an SI file (see page 12-8: COPYRM; CLIB/P: COPYBLK).

** 7-track and 9-track Tapes **

The same physical tape may be written as 7-track at one time and as 9-track at another. Pages 1-3, 1-4, list the tape drives available on each mainframe. The job card must specify separately the maximum number of 7-track (MT) and 9-track (GE/PE/HD) tape drives which will be needed simultaneously in the job. The density parameter on the LABEL or REQUEST statement informs the system of the type of each individual tape. (See page 11-11: 'Labelled Tapes' and page 11-13: 'Unlabelled Tapes'.)

Nine-track tapes can be used for anything that 7-track, SI or stranger, can, except:

- . 9-track lowest density is 800 bpi.

Nine-track tapes have the following capabilities not available on 7-track:

- . 9-track labels can be ASCII or EBCDIC (default: ASCII).
- . 9-track stranger tapes can be coded ASCII or EBCDIC.
- . 9-track tapes can be 1600 bpi or 6250 bpi.

On 9-track coded stranger tapes, a 6-bit character in memory is converted to or from an 8-bit character on tape. On 9-track SI or binary stranger tapes, four 6-bit units in memory become three 8-bit units on tape, with no conversion (sometimes called packed mode). All 9-track tapes are odd parity, while 7-track tapes can be even (coded) or odd (binary). Other Computer Center equipment which handles 9-track tapes includes the Datagraphix Autocom Recorder (6250 and 1600 bpi only) (page 23-1) and Xerox-8700 (1600 bpi only) (page 23-7). Other Computer Center equipment which handles 7-track tapes includes the Calcomp model 936 plotter.

** FORTRAN Tapes **

FORTRAN coded (unit) records have a default maximum of 150 characters, which are buffered into the 128-word PRU. FORTRAN binary records are blocked into the 512-word PRU, but FORTRAN write statements are not limited to 512 words. FORTRAN may create stranger tapes for other equipment by using:

- a) 'BUFFER OUT' statements (to avoid having NOS/BE level numbers);
- b) Formatted write with tape blocking characteristics defined on FILE and LDSET statements (see pages 5-11, 8-6).

** Considerations for Creating Tapes for Data Interchange **

When writing a tape on a DTNSRDC CDC computer for later use as input to a non-CDC computer system, the user should be aware of potential problems resulting from write errors.

NOS/BE normally employs a very effective method of write-error recovery. When an "unwritable" spot is encountered while writing a tape, NOS/BE writes a unique 24-bit "noise bracket" record prior to the bad spot, locates the next good portion of the tape and writes a matching noise bracket record. Each bad spot is, therefore, preceded and followed by a noise bracket. When this tape is read by CDC CYBER systems, NOS/BE recognizes the noise brackets and positions the tape past each bad spot, thereby avoiding a read error.

Non-CDC systems do not recognize CDC noise bracket records as such and may be unable to read a tape which contains them. For this reason, a user creating a tape for system interchange should inhibit noise bracket generation by adding the IB parameter to the LABEL or REQUEST control statement (see NOSBE, 4-52, 4-67 and 4-69). The IB parameter is not required for 9-track tapes recorded at 6250 or 1600 bpi since the system will automatically inhibit noise brackets for them. When a write-error occurs on a tape for which noise brackets have been inhibited, NOS/BE erases six inches of tape and retries the write in the new position. This erase/rewrite sequence is repeated, if unsuccessful, up to a maximum of 50 times, after which the console operator is given the option of retrying recovery or aborting the job. It is possible, therefore, for a tape to contain up to 25 feet of erased noise between good blocks. While the CYBER system should be able to read such a tape successfully, the results of reading it on a non-CDC system would depend on that system's read-error recovery capability.

It is recommended that:

- a. Noise-bracket generation should be inhibited when writing a tape (other than 9-track 6250 or 1600 bpi) destined for a non-CDC system.
- b. If a recovered write error occurs on an interchange tape (entered in the dayfile as "WRT RVD, PRU=xxxxxxx"), the tape should be recreated or copied to eliminate the error.

For stranger tapes (page 11-1) being prepared for use on other computers, blocking is recommended when the destination computer supports blocking. The greater the blocking factor (number of records per block), the greater the number of records which can be put onto a tape. The following table illustrates the effect of different blocking factors (RB) on a file of card images written on an 6250-bpi, 9-track 2000' tape:

rb	ipb	bpt	cds/tape
1	.31	77,419	77,419
2	.33	72,727	145,454
20	.56	42,857	857,140
40	.81	27,906	1,116,240
100	1.58	15,384	1,538,400
1000	13.10	1,832	1,832,000

Similarly, on a 800-bpi, 7-track coded 2000' tape:

rb	ipb	bpt	cds/tape
1	.85	28,235	28,235
2	.95	25,263	50,526
20	2.75	8,727	174,540
40	4.75	5,052	202,080
100	10.75	2,232	223,200
1000	100.75	238	238,000

To estimate the number of SI tape blocks a disk file will require:

$$\text{bpt} = \text{diskprus} / 2 \quad (\text{coded})$$

$$\text{bpt} = \text{diskprus} / 8 \quad (\text{binary})$$

where diskprus is the number of PRUs as indicated in the CATALOG (CT) or EXTEND (EX) messages or in an audit.

To estimate the number of stranger tapes (nst) required to hold a number of fixed-length records:

$$\text{nst} = \frac{\text{nrecs}}{\text{tapelength}} \left[\frac{\text{reclength}}{\text{density}} + \frac{\text{ibg}}{\text{rpb}} \right]$$

where nrecs is the number of records
 reclength is the record length in characters (bytes)
 rpb is the number of records per block (blocking factor)
 tapelength, density, ibg are described on page 11-4.

*** Magnetic Tape Formats ***

If you are bringing programs or data on magnetic tape to the Computer Center, the following tape formats are suggested to ease data transfer and translation.

machine	tracks	label	record format	#char /rec	block size	density (bpi)	character code	parity
IBM	9	1bld	fixed block	80*	<3840	1600	EBCDIC	odd
IBM	7	1bld	fixed block	80*	<5120	800	BCD	even
CDC	9	1bld	fixed block	80*	<3840	1600	EBCDIC	odd
CDC	7	1bld	fixed block	80*	<5120	800	BCD	even
Univac	9	un1bld	fixed block	80*	3840	800	EBCDIC	odd
Univac	7	un1bld	fixed block	80*	4800	800	BCD	even
DEC	9	un1bld	fixed block	80*	4800	800	ASCII	odd
Burroughs	9	un1bld	fixed block	80*	3840	1600	EBCDIC	odd

* for source programs. Data may be any length, preferably <=150.

For all tapes, the above information is required.

All tapes should be accompanied by a summary of the contents and a tape format listing. If you have questions about the acceptability of a particular tape format or code, call User Services ((202) 227-1907) before sending a tape to the Computer Center. See Form 277 (12-77) FIPS Pub. 53.

If imported data or programs have fewer than 500 card images, punched cards may be used instead of a tape. Though cards cost more to transport, they avoid many frustrations and complications.

An imported tape should first be copied onto disk or to another tape to convert it to CDC internal format. Card decks should be read once and cataloged on disk for later use.

To send tapes from the Computer Center to another installation, you should request from the receiving site its preferred tape formats and then contact User Services for assistance in writing the tape.

*** Tape Assignment ***

Two classes of tape storage are provided in the Computer Center, 'Diebold' and 'slot'. Tapes which are used frequently should be permanently stored at the proper computer (i.e., CYBER 176 or CYBER 750) in the Diebold cabinets, which are mechanized storage units accessible to the computer operators. These tapes are assigned a permanent external label indicating their location by cabinet, shelf and position, such as 'CB2499', and are referred to as 'Diebold' tapes. The VSN of a Diebold tape is the same as the external label.

Tapes which are seldom used on the CDC CYBER computers, which are being transferred between systems, or which are normally retained by the owner are assigned a temporary slot number for up to a 24-hour period at the ADP Control Center of the computer on which they are to be used. At the end of the day's processing (or earlier at the user's request), these are returned to the user and will require a new slot number assignment for the next use.

VSN for a slot tape is 'SLOTxx=id'

where xx is the assigned slot

id is the user's external sticker on the tape reel
(six (6) one-inch-high characters, please, for easy
reading by the operator)

Tapes belonging to remote users may be sent to the Tape Librarian. Special slots (A1-A9, B1-B9, 71-84) will be assigned for up to two weeks.

All tapes to be called for in a job must be supplied by the user as Diebold tapes and/or slot tapes. No scratch tapes are available.

Tapes stocked by the Computer Center are of 2400-foot nominal length (10.5 in. diameter). Smaller tapes may be used. For assignment of tapes to Diebold locations or temporary slots, to have a tape moved from one computer room to another, and to arrange for the purchase of tapes, see the Tape Librarian, Code 1896, (202) 227-1227.

*** Tape Care and Cleaning ***

Tapes should be stored in closed containers in racks which give them vertical support. Tapes may not be spliced. They should be read and rewound at least every six months. Logs should be kept on contents, format, and creation dates of tapes (an interactive program 'DOCS' may be used for this purpose (see CLIB/U)).

If a tape has many parity errors, cleaning it may help. Even a brand new tape may need cleaning. This off-line process does not destroy the information on the tape. If a tape receives heavy usage, cleaning it after ten or more uses may reduce the incidence of parity errors. The Tape Librarian can also perform tape certification, which determines whether there are any areas on the tape which do not record properly. Certification DESTROYS current information on the tape (except VSN). To change the VSN, contact the Tape Librarian and request blank labelling or degaussing.

If, after a tape has been cleaned, it still has many parity errors, it should be exchanged for a new tape. The information on the old tape is not recovered automatically in this case.

To have a tape cleaned or certified, submit an off-line work request to the Tape Librarian. Users who are not at the Carderock site should call (202) 227-1227.

When possible, slot tapes should be in the Computer Room environment for at least two hours before reading or writing. This allows temperature and humidity to stabilize and should minimize tape problems.

Please notify Code 1892.1, (202) 227-1907, of any unusual tape problems.

*** VSN ***

All jobs which use magnetic tape must identify the reel(s) by the VSN control statement or the 'VSN=' parameter on the LABEL or REQUEST statement. The VSN statement must precede the LABEL or REQUEST statement for the file. A single VSN statement can contain declarations for several reels and files. If a local file name is reused by a different reel during one job, the UNLOAD statement must precede the VSN control statement for the second reel. To be compatible with other computers, each volume serial number is six (6) characters.

VSN(lfni=vsni,lfni2=vsni2...) Diebold tapes, unrelated lfni's.
vsni,2,... are physical storage ID's.

VSN(lfni=vsni/vsni2/...) Diebold tapes, multi-reel file.
vsni,2,... are physical storage ID's.

VSN(lfni=vsni3=vsni4) Slot tape or other reel where
physical storage ID is not same as tape ID at original
write. By local convention, vsni3 is slot number or
physical storage ID; vsni4 is the external sticker ID and
is written internally on the tape the first time a label
is written.

VSN(lfni=vsni1=vsni2/vsni3=vsni4/vsni5=vsni6) multi-reel slot tapes
lfni is local file name on a subsequent LABEL or
REQUEST control statement.
vsni is volume serial number (e.g., CA0599 or SLOT88 or
ABCDEF) six alphanumeric characters. Each tape is
identified by two vsni.

Be extremely careful of 0 (zero) and O (oh) characters as a computer
program checks this label ID.

** Examples **

(LABEL and REQUEST statements are explained later)

1. Single reel files (Diebold tapes):

jobname,GE2,... (2 9-track tapes used at same time)

...
LABEL,OLDPL,L=xxxxLIBR,R,D=GE,NORING,VSN=CA0299.

LABEL,NEWPL,L=xxxxLIBR1,W,D=GE,RING,VSN=CA0298.

2. 3-reel file (Diebold tapes):

jobname,PE1,... (only one tape in use at a time)

...
PAUSE. multi-reel file reading CA0388/CA0485/CB1095.

VSN,MISFILE=CA0388/CA0485/CB1095.

LABEL,MISFILE,L=xxxxFY84DATA,R,D=PE,NORING.

3. Slot tape with sticker ID SGX402, unlabelled, stranger:

jobname,MT1,....

...

VSN(TAPE55=SLOT14=SGX402)

<-- SLOTx precedes sticker ID

...

REQUEST,TAPE55,HY,S,RING.

4. 2-reel file of slot tapes:

jobname,HD1,.... (9-track tape, only one in use at a time)

...

PAUSE. multi-reel file writing SLOT02/SLOT19.

VSN,TAPE9=SLOT02=PUVZ01/SLOT19=PUVZ02.

...

LABEL,TAPE9,L=PUVZLONG,W,D=HD,RING.

5. Labelled slot tape where label and sticker ID differ
(<volser> is the volume serial number in the tape label record;
<stkrid> is the sticker label on the outside of the tape):

jobname,GE1,....

...

VSN(TAPE73=SLOT05=<stkrid>=<volser>)

...

LABEL,TAPE73,L=xxxxOLDPL,R,D=GE,NORING.

6. Labelled Diebold tape where label on tape differs from Diebold storage location:
(In this example, the tape label was originally written with a letter oh (O) instead of a digit zero (0).)

jobname,MT1,....

...

VSN(TAPE82=CA0199=CA0199)

...

LABEL,TAPE82,L=xxxxMYTAPE,D=HY,....

7. Write on an unexpired tape whose label was written on Dec 31, 1982:

jobname,HD1,....

...

LABEL,TAPE10,L=xxxxPLOT TAPE,W,RING,D=HD,C=82365,....

*** Labelled Tapes ***

All tapes should be labelled. Labelled tapes provide some safeguards against using the wrong tape, and require less operator action when they are mounted. Label information is magnetically recorded on the tape. Labels are ANSI format. The LABEL statement is used to describe the tape.

LABEL(lfn,L=xxxxn,C=yyddd,D=z,E=z,F=z,IB,N=z,noring,r,T=z,VSN=vrno)

lfn is the local file name (same as on other statements referencing the file).

L=xxxxn is file label name up to 17 characters. Usually, first four should be xxxx from CHARGE card. (Required if R and RING are specified.)

C= Creation date. Required if label is unexpired and W is specified. (see 3-11: Example 7)

D= Tape and label density.
GE 6250 bpi 9-track tape (group encoded)
PE 1600 bpi 9-track tape (phase encoded)
HD 800 bpi 9-track tape (high density)
HY 800 bpi 7-track tape (hyper density)
HI 556 bpi 7-track tape (high density)
LO 200 bpi 7-track tape (low density)

E= Edition number (defaults: CDC NOS/BE (level 508 on), B7700, IBM, etc.: E=00; NOS/BE (before level 508): E=01)

F= Tape format. Omitted for SI tape.
S stranger tape (for offline device or other site)
L long stranger tape (blocks longer than 512 words)

IB If specified, inhibit CDC noise brackets on non-PE tapes. Use this when writing tapes for other computers (B7700, IBM, Calcomp, COM)

N= Code for 9-track conversion (label and/or coded stranger tapes)

EB - EBCDIC

US - ASCII (default)

A label written with N=EB must be read with N=EB.

noring is RING if tape is to be written
(L= required if R is specified)
NORING if tape is to be read only (default)

r is W when writing tape label using supplied parameters.
 C= is required if existing label is unexpired.
 R when reading label, will check against supplied
 parameters. L= is required if RING is specified.
 (This parameter has no default.)

T= Retention period in days (1 to 3 digits) (default: T=1)
 System will not allow W (rewrite label) if label
 retention has not expired, unless C= is also specified.

VSN=vrno 6-character visual reel number (cabinet, shelf, number)
 for internal checking. Optional if VSN statement is used.
 Must use separate VSN statement for multi-reel or slot
 tapes.

Comments to the operator may be written between columns 50 and 70.

See page 11-9: Examples; NOSBE, 4-50.

A labelled tape may contain more than one labelled file for
compatibility with other computer systems. See page 11-14; NOSBE,
3-37: Multi- file Set.

*** Unlabelled Tapes ***

Unlabelled tapes should be avoided, if possible. They are often used to transfer files between the CDC computers and other hardware. Multi-reel files are not defined for unlabelled stranger tapes. An unlabelled tape is described by a REQUEST statement instead of a LABEL statement.

REQUEST, lfn, dt, eb, noring, type, IB, VSN=vrno.

lfn is the logical file name (same as on other statements
 referencing the file)

dt is the density
 GE - 6250 bpi 9-track tape (group encoded)
 PE - 1600 bpi 9-track tape (phase encoded)
 HD - 800 bpi 9-track tape (high density)
 HY - 800 bpi 7-track tape (hyper density)
 HI - 556 bpi 7-track tape (high density)
 LO - 200 bpi 7-track tape (low density)

eb is omitted for 7-track
 omitted or US for 9-track ASCII format coded
 EB for 9-track EBCDIC format coded

noring is RING if tape is to be written
 NORING if tape is to be read only (default)

type is the data format
 omitted - SI
 S - a stranger tape with physical record
 size not exceeding 512 words.
 L - a stranger tape with records over 512
 words.

IB If specified, inhibit CDC noise brackets on non-PE tapes.
 Use this when writing tapes for other computers (B7700,
 IBM, Calcomp, COM)

VSN=vrno 6-character visual reel number (cabinet, shelf,
 number) for internal checking. Required unless VSN
 statement is used (must use separate VSN statement for
 slot or multi-reel tapes).

Comments to the operator may be written between col 50 and col 70.

See page 11-10: Example 3.

*** IBM-style Multi-file Labelled Tapes ***

SI labelled tapes normally contain one label file of two 80-character records, one or more files of data (programs, etc.), and a terminating trailer file of one 80-character record.

Some stranger tapes (IBM, VAX and B7700) label each data set on the reel with a unique header and trailer label file, so that a reel of three (3) data files would contain nine (9) physical files. See NOSBE, 3-37: Multi-file Set.

To read such a multi-file reel on CDC:

```
VSN,reel=SLOTxx=visnum.  
COMMENT. MF,E parameters for existing multi-file reel  
REQUEST,reel,S,EB,NORING,PE,MF,E.  
COMMENT. locate specific data file by label name  
LABEL,file1,L=$namex.fileno$,D=PE,F=S,N=EB,R,M=reel.  
...  
COMMENT. locate specific data file by position  
LABEL,file4,D=PE,F=S,N=EB,R,M=reel,P=4.  
...
```

To list the set of labels on a multi-file reel:

```
VSN,SET1=SLOTxx=visnum.  
REQUEST,SET1,E,EB,NORING,PE,S,MF.  
LISTMF,MF=SET1,P=1.
```

Note: In above examples, \$...\$ is used to enclose any name which includes special characters.

Local file name on VSN and REQUEST is multi-file name for M=.

Tapes from VAX/VMS might also require ',E=nn' on the LABEL statement, where <nn> is the version number-1 of the original VAX disk file from which the tape was made.

***** UTILITIES *****

A wide variety of utility programs is available as a part of the system or in libraries or other files. This chapter describes many of the system utilities (no ATTACH required). See CCRM, chapter 19; CLIB; CLIB/U for additional utilities.

Utility operations can be performed with named files, each assigned to a specific peripheral device (may need REQUEST statements which must precede the utility control statements). Most copy utilities use a field length of 20000. All NOS/BE utilities described below are for sequential files only. SIS files are manipulated by COPYBF or FORM. For random files, see FORM; SEQTORAN directive of EDITLIB (page 17-4); A and B options of UPDATE (page 16-2); COPYE (page 12-6); COPYS (CLIB/U); COPYBFR (page 19-3). COPYRM (page 12-8) may be used to copy most files.

*** NOS/BE utilities ***

See NOSBE, chapter 4, for a full description of the following utilities.

COPYBF(infyl,outfyl,nfiles)	copy binary files
COPYCF(infyl,outfyl,nfiles)	copy coded files
COPYBR(infyl,outfyl,nrecs)	copy binary records
COPYCR(infyl,outfyl,nrecs)	copy coded records

infyl - input file (default: INPUT)
outfyl - output file (default: OUTPUT)
nfiles - number of files (default: 1)
nrecs - number of records (default: 1)

Any tape parity will request intervention from the operator.

Defaults: COPYBF(INPUT,OUTPUT,1)
COPYCF(INPUT,OUTPUT,1)
COPYBR(INPUT,OUTPUT,1)
COPYCR(INPUT,OUTPUT,1)

SKIPF(file,n,level,type) skip forward
SKIPB(file,n,level,type) skip backward

file - name of file to be positioned
n - number of records/files to be skipped
level - octal end-of-record level number
 0 or omitted - normal end-of-record
 17 - end-of-file
type - one of:
 C - coded files (for 7-track, even parity)
 B - binary files (for 7-track, odd parity)

SKIPF and SKIPB may not be used for S or L tapes (NOSBE, 4-84, 4-85).

Defaults: SKIPF(FILE,1,0,B)
 SKIPB(FILE,1,0,B)

COPYL(oldfyl,binary,newfyl) selective replacement of routines
COPYL(oldfyl,binary,newfyl,last,flag)

COPYLM(oldfyl,binary,newfyl)
COPYLM(oldfyl,binary,newfyl,last,flag)

oldfyl - old master binary (default: OLD)
binary - replacement file (default: LGO)
newfyl - create updated master file (default: NEW)
last - name of last record on <oldfyl> to be processed.
 (default: all records on <oldfyl> are processed)
flag - processing options
 R - rewind <oldfyl> and <newfyl> before processing
 (<binary> is always rewound before and after)
 A - append to end of <newfyl> all binary records that
 did not match any of <oldfyl>. To use both R and A
 options, combine the letters "RA".
 (default: option not selected)

Defaults: COPYL(OLD,LGO,NEW)
 COPYLM(OLD,LGO,NEW)

COPYL processes the master file (<oldfyl>) forward only, but will search <binary> and replace all routines of the same name from their first occurrence on <oldfyl>.

COPYLM differs only in its handling of multiple occurrences of a routine on the old master. Using COPYLM, all occurrences of the routine on <oldfyl> are replaced by the first matching record on <binary>. This facilitates replacing a subroutine which occurs in several overlays. (See NOSBE, page 4-22.)

COPYN(f1,outf1,infy1,...,infyn) copy, merge or select logical
records from up to 10 binary
input files

f1 - the record format; use 0 (zero).
outf1 - the output file (precedes all input files).
infy1 - the input file(s)

The text logical record, which contains record identification cards, will be file INPUT. The command summary will be in file OUTPUT. If run interactively, files INPUT and OUTPUT may not be connected to the terminal, nor can they be replaced by other file names.

The record identification cards begin in column 1 and have the following format:

p1,p2,p3

where p1 is a record name or number, relative to current position,
to start the copy.
p2 is the last record name of a set or total number of records
to be copied.
if p2 is omitted, only p1 is copied.
if p2 is "*", the copy is thru end-of-file.
p3 is the name of the input file to search (may be omitted).

Record identification cards must be blank (thru column 80) following the last specified parameter.

Several file manipulation commands are also allowed.

When possible, search in the order the routines exist in the file. COPYN will do an end-around search, so it is costly if many rewinds are required.

Default: no defaults.

ITEMIZE(lfn) list contents of binary file
ITEMIZE(lfn,<parameters>)

Output includes record number, name, length, prefix table for relocatable binary or user library. For sequential pl, only deck names are listed. (See page 17-4: CONTENT; CLIB/U: LISTBIN; NOSBE, 4-59)

Optional parameters include:

- E - list entry points for relocatable programs or idents for
 UPDATE sequential PL
- L=lfn - if other than "OUTPUT"
- N - itemize to end-of-information
- N=nn - itemize nn files (default: 1)
- PW=nn - page width
 (batch default: 136/72; if nn<136, 72 is used,
 if nn>136, 136 is used)
 (Intercom default: 136 (if output file not connected);
 72 (if output file connected))

Default: ITEMIZE(LGO,L=OUTPUT,N=1,PW=<see above>)

BKSP, lfn.

Backspace file ready to read last previous logical record. If file was after an EOF, it will be positioned just before the EOF.

Default: no defaults.

COMBINE, file1, file2, n.

Concatenate the first <n> logical records on <file1>, making one logical record on <file2>. Use n=0 to combine thru EOF.

(Does not work with FORTRAN unformatted files.)

Default: no defaults.

(See also page 12-9: COPYSF)

COMPARE, file1, file2.

Verify that a copy is error-free.

Compares records from current position with additional optional parameters to control number of records and number of errors.

Note: Disk files should not be compared with tape files using this utility because erroneous 'bad compare's can occur.

Default: no defaults.

(For a better routine for comparing text files, see page 12-10: COMPAR)

COPY, file1, file2.

Copy to double end-of-file (or end-of-information, if encountered first). Tape input or output is binary.

Default: COPY, INPUT, OUTPUT.

(For a better routine, see page 12-6: COPYE)

*** Other Utilities ***

The following utilities were obtained from the University of Washington and have been made a part of the DTNSRDC CDC NOS/BE computer operating systems (no ATTACH required): COPYE, COPYF, COPYR, COPYRM, COPYSF/COPYSBF/COPYSR (which replaces and expands the CDC version of COPYSBF). Full documentation may be obtained by using the following:

BEGIN,DOCGET,,OTHER,,*,OUTPUT,MSACCES=<pw>.

where * is one of: COPYE, COPYF, COPYR, COPYRM, COPYSF.

To print several documents, use:

BEGIN,DOCGET,,OTHER,,,OUTPUT,I=dirfyl,MSACCES=<pw>.

where file 'dirfyl' contains the desired document names, comma-separated on as many lines as desired.

COPYE(infyl,outfyl)

Make an exact copy from current position
to end-of-information.

If <infyl> is random and if <infyl> and <outfyl> are disk and are rewound, then <outfyl> is random.

Mode switching will occur if a parity error occurs in the first PRU of <infyl>.

This is the easiest way to produce an exact disk copy of an arbitrary NOS/BE file.

This is the fastest way to copy disk-to-disk.

On tape (not recommended), <outfyl> is binary.

Default: no defaults.

COPYF(infyl,outfyl,nfiles)	Copy nfiles files
COPYF(infyl,outfyl,nfiles,C)	Copy nfiles coded files
COPYR(infyl,outfyl,nrecs,level)	Copy nrecs records
COPYR(infyl,outfyl,nrecs,level,C)	Copy nrecs coded records
COPYF(,outfyl,neofs)	Write neofs eof's
COPYF(,outfyl,neofs,C)	Write neofs coded eof's
COPYR(,outfyl,neors,level)	Write neors eor's
COPYR(,outfyl,neors,level,C)	Write neors coded eor's

infyl - input file (no default)
 outfyl - output file (no default)
 n - number of files/records to copy or
 number of eof/eor to write
 (default: 1)
 C - copy coded files/records or
 write coded eof/eor
 level - eor level number (octal)
 (default: 0)
 (COPYR only. If used, neors must also be used.)

NOS/BE files only; no S or L tapes.

For COPYF, if eoi is encountered without eof just before it, an eof is added to <outfyl>.

For COPYR, if last record written was of a level less than the specified level, a zero-length record of the specified level is written.

For COPYR, a record level may not be specified without a record count.

Unless FILE statements are used, any tape parity error is immediately fatal.

Default: no default file names. Other defaults as listed above.

Messages

BCD MODE FILE m	COPYF/R	tape mode switched to coded
BIN MODE FILE m	COPYF/R	tape mode switched to binary
EOF ENCOUNTERED	COPYR	COPYR stops
EOI ENCOUNTERED	COPYR	COPYR stops
EOI ENCOUNTERED-m FILES	COPYF	m-th file ended with eof, followed by eoi. COPYF stops.
EOI ENCOUNTERED-FILE m	COPYF	m-th file ended with eoi, without eof. Eof added to outfyl. COPYF stops.

COPYRM(infyl,outfyl,n)

Copy and convert records on sequential (SQ) files from one record type and block structure to another.

infyl - input file (no default)

outfyl - output file (no default)

n - a count which may be expressed in any of the following forms (default: 1F):

EOI - copy to end-of-information (SI files only)

mF - copy m files (partitions)

mP - same as mF

mR - copy m system logical records (sections)

mS - same as mR

m - same as mF

FILE statements are used to specify conversion and blocking. (See page 5-11)

COPYRM is especially convenient for reading and writing blocked stranger tapes (both 7-track and 9-track).

COPYRM is fast for tape and other conversion copies.

See also procedure COPYBLK (CLIB/P).

COPYSF(infyl,outfyl,n,params) Copy (shifted) files
 (synonym: COPYSBF)
COPYSR(infyl,outfyl,n,params) Copy (shifted) records

infyl - input file (must be first parameter)
 (default: INPUT)
outfyl - output file (must be second parameter)
 (default: OUTPUT)
n - number of files/records to be copied (default: 1)
 (Usually third parameter, but may appear elsewhere.)
params - other parameters and options which may follow in any
 order. Some of them are:
 Sm - shift m columns to right (0-132) (default: 1)
 C - initial record is coded
 (automatic parity switching occurs, if
 necessary, at start of each logical record on
 tape files.)
 COMB - remove all embedded eor/eof's
 EOI - copy to eoi for COPYSF/COPYSBF
 EOF - copy to eof for COPYSR
 LC=1 - line count, <1> is line limit (default: 2000)
 LR - list end-of-record and end-of-file as *EOR and
 *EOF, respectively. (default: LR not selected)
 L8 - select 8 lines per inch output
 (restored to 6 lines per inch at end)

This is an adaptation of the standard CDC utility COPYSBF with many additional options and capabilities. It is used primarily for listing files but can be used as a general purpose utility.

Defaults: COPYSF(INPUT,OUTPUT,1,S1,LC=2000)
 COPYSR(INPUT,OUTPUT,1,S1,LC=2000)

For a copy of the full documentation, use

BEGIN,DOCGET,,OTHER,,COPYSF,OUTPUT,MSACCES=<pw>.

The following utility was obtained from the University of Minnesota.

COMPAR,filea,fileb,output,modfyl/options. Compare two text files
and report any differences

filea - first file to be compared
fileb - second file to be compared
output - listable output file to contain the report of the compare
modfyl - will contain update directives to convert filea to fileb
options - control for the comparison (see document)

Default: COMPAR,FILEA,FILEB,OUTPUT,MODS/C6,D,W120.

COMPAR is designed for text files of up to 150 characters per line,
with trailing blanks ignored.

The NOS/BE COMPARE utility (page 12-3) will compare any files.

To speed up COMPAR, the W option should be used when the lines
being compared are shorter than 120 characters (the default).

COMPAR is in library UTILITY. To execute:

ATTACH,UTILITY.
LIBRARY,UTILITY.
COMPAR,<params-and-options>.

For a copy of the document, use

BEGIN,DOCGET,,UTILITY,,COMPAR,OUTPUT,MSACCES=<pw>.

*** Sample Utility Setups ***

** Copy BCD from File 3 to New Tape, File 1 **

```
xxxxTAP,GE2.                                name / code
CHARGE,....
LABEL,SGMAST,L=xxxxMASTER,W,D=GE,T=7,VSN=CA7777,RING.
LABEL,SGSHIP,L=xxxxSHIPS,R,D=GE,VSN=CA8888,NORING.
COPYF(SGSHIP,NULL,2,C)      <-- copy first 2 files to position tape
RETURN,NULL.
COPYF(SGSHIP,SGMAST,C)      <-- copy third file to a new tape
REWIND(SGSHIP,SGMAST)      <-- will print block count written
<eoi>
```

A better way to skip ahead two files:

SKIPF(SGSHIP,2,17,C) instead of COPYF(SGSHIP,NULL,2)

** Multiple Input to FORTRAN Program **

```
xxxxTP3,CM120000,T200.                    name / code
CHARGE,....
COPYR(INPUT,TAPE2)      <-- copy cards to file called TAPE2
REWIND,TAPE2.           <-- reposition TAPE2 for reading
FTN5.
LGO.
<eor>
      (data to be read from TAPE2 in program)
<eor>
      PROGRAM MANIP (INPUT, OUTPUT, TAPE2, TAPE5=INPUT, TAPE6=OUTPUT)
      (FORTRAN source cards)
<eor>
      (data to be read from TAPE5)
<eoi>
```

** Tape-to-Print - Single Space **

```
xxxxPR,GE1.                                name / code
CHARGE,....
LABEL,TAPE9,L=xxxxSOURCE,D=GE,R,VSN=CA9999,NORING.
COPYSF,TAPE9,OUTPUT,C.      <-- copy shifted coded file
<eoi>
```

** Replace Subroutines in Relocatable Binary Program **

```
jobname,.... name/code
CHARGE,....
FTN5 (B=BINARY,OPT=1)
REQUEST,NEWLIB,*PF.
ATTACH,OLDLIB,PROGBIN,ID=xxxx,PW=RDONLY. <-- see page 13-24 for create
REWIND,BINARY.
COPYL(OLDLIB,BINARY,NEWLIB)
NEWLIB. <-- execute corrected file
CATALOG,NEWLIB,PROGBIN,ID=xxxx,PW=RDONLY,XR=RDONLY.
PURGE,OLDLIB. <-- after catalog of new cycle, purge old one
<eor>
      (subroutines to replace old version)
<eor>
      (data)
<eoi>
```

** Replace, Insert or Delete Routines on Binary Object File **

```
...cards to set up the three files
COPYN(0,NEWLIB,OLDLIB,BINARY)
<eor>
DECK1,DECKL,OLDLIB <-- copy first 1 decks from oldbin file
PUTIN,,BINARY <-- insert new routine PUTIN
DECKM,DECKN,OLDLIB <-- copy last 2 decks from oldbin file
ADDON,*,BINARY <-- add on rest of new routines from binary
<eoi>
```

** Select Routines from Binary (Object) File **

```
jobname,.... name/code
CHARGE,....
ATTACH,BINLIB,ID=xxxx,MR=1.
REQUEST,GOFIL,*PF.
COPYN(0,GOFIL,BINLIB)
COMMENT. two alternative ways of selecting a routine are shown.
CATALOG,GOFIL,ID=xxxx,XR=VOIG.
<eor>
EIGEN
HESS2,,BINLIB
WEOF(GOFIL)
<eoi>
```

** Copy a Random File **

```
jobname,.... name/code
CHARGE,....
ATTACH,RANIN,.... <-- input random file
REQUEST,RANOUT,*PF. <-- output file
COPYE,RANIN,RANOUT.
CATALOG,RANOUT,....
<eoi>
```

** Convert Blocked Stranger Tape (7-track) **
** to Card Image Disk File **

```
jobname,MT1,.... name/code
CHARGE,....
VSN,TAPE=SLOT06=MYTAP1.
REQUEST,TAPE,HY,S,NORING.
REQUEST,DISK,*PF.
FILE,TAPE,BT=K,RT=F,MBL=5120,CM=YES,MRL=80,RB=64. <-- input
FILE,DISK,BT=C,RT=Z,MRL=640. <-- output
COPYRM,TAPE,DISK.
CATALOG,DISK,....
<eoi>
```

** Convert 9-track EBCDIC Tape File to Card Image Disk File **

```
jobname,PE1,.... name/code
CHARGE,....
VSN,TAPE=SLOT03=MYTAP2.
REQUEST,TAPE,PE,S,EB,NORING.
REQUEST,DISK,*PF.
COMMENT. open with rewind (OF=R), close with unload (CF=U) <-- optional
FILE,TAPE,BT=K,RT=F,MRL=80,MBL=5120,RB=64,CM=YES,OF=R,CF=U.<-- input
COMMENT. rewind before and after use <-- optional
FILE,DISK,BT=C,RT=Z,MRL=640,OF=R,CF=R. <-- output
COPYRM,TAPE,DISK.
CATALOG,DISK,....
<eoi>
```


** Create 9-track ASCII Tape of Card Images Blocked 20 **

```
jobname,HD1,....                      name/code
CHARGE,....
VSN,TAPE=SLOT04=MYTAP3.
ATTACH,DISK,CARDIMAGES,ID=xxxx.
LABEL,TAPE,L=label,D=HD,F=S,N=US,RING.
FILE,DISK,BT=C,RT=Z,MRL=640.          <-- input
FILE,TAPE,BT=K,RT=F,MRL=80,RB=20,MBL=1600,CM=YES.  <-- output
COPYRM,DISK,TAPE.
RETURN,TAPE.
<eoi>
```

** Copy 9-track EBCDIC Stranger Tape to Disk **
(medium sized records)

```
jobname,PE1,....                      name/code
CHARGE,....
REQUEST,TAPE,PE,NORING,S,EB.
REQUEST,DISK,*PF.
FILE,TAPE,BT=K,RT=F,MRL=982,MBL=2950,RB=3,CM=YES.  <-- input
FILE,DISK,BT=C,RT=Z,MRL=1280.          <-- output
COPYRM,TAPE,DISK.
CATALOG,DISK,MYFILE,ID=xxxx.
<eoi>
```

***** FORTRAN EXTENDED *****

*** Introduction ***

NOS/BE at DTNSRDC has three FORTRAN compilers:

- FTN5 - FORTRAN 77 Extended, Version 5 (ANSI 1978 standard)
- FTN4 - FORTRAN IV Extended, Version 4 (ANSI 1966 standard)
(also called FTN)
- RUN - an obsolete FORTRAN no longer supported by CDC or
DTNSRDC (it is not discussed further in this manual)

For details of the FTN5 language, see pages 13-2 ff and page i:
FTN5.

For details of the FTN4 language, see pages 13-30 ff and page i:
FTN4.

Two additional FORTRAN products are also available:

- RATFOR - a pre-compiler for FTN4
- MNF - Minnesota FORTRAN (extremely good diagnostics)

These are described briefly in this chapter.

*** FTN5 Control Statement Parameters ***
(see FTN5, chapter 11)

Examples FTN5. defaults to (ANSI=0,B=LGO,BL=0,CS=FIXED,DB=ER,DO=0,
DS=0,E=OUTPUT,EL=T,GO=0,I=INPUT,L=OUTPUT,LO=S/A,
OPT=0,PD=6,PL=5000,PN=0,PS=60,PW=136,QC=0,REW=0,
ROUND=A/S/M/D,SEQ=0)
FTN5(GO) compile and execute
FTN5,I=MYFILE,ROUND,OPT=1.
FTN5(B=PUNCHB,EL=T,ANSI)

option	action
ANSI=T	Non-ANSI usage flagged but considered trivial.
ANSI=F	Non-ANSI usage flagged and considered fatal.
ANSI	Same as ANSI=T.
ANSI=0	No ANSI diagnostics generated.
omitted	Same as ANSI=0.
B=PUNCHB	Produce punched binary decks of all routines; no BCD sequencing in the last columns of the cards.
B=lfm	Put binary images on file <lfm>.
B	Same as B=BIN.
B=0	No binary output.
omitted	Same as B=LGO.
BL	Burstable list. Each major section of compilation listing starts on a new page.
BL=0	Compact list. New page for first page only. (default)
omitted	Same as BL=0.
CS=FIXED	Collating sequence fixed weight table (display code).
CS=USER	Weight table is user-defined by subroutines COLSEQ, WTSET, CSOWN. (FTN5, 7-29)
CS	Same as CS=USER (at DTNSRDC).
omitted	Same as CS=FIXED (at DTNSRDC).

option	action
DB=op/op/...	Debugging options. (ARG=FIXED may not be specified) <op> is one of: ER - generate code for object-time reprieve of errors ID - generate output for interactive debug (requires OPT=0) PMD - post mortem dump facility is used SB - check subscript bounds SL - check character substring expressions ST - same as DB=ID but no stylized object code TB - full error traceback
DB	Same as DB=ER/ID/PMD/SB/SL/ST/TB.
DB=0	All options deselected (DB=-ER/-ID/-PMD/-SB/-SL/-ST/-TB).
omitted	Same as DB=0 (if OPT=1,2,3). Same as DB=ER (if OPT=0).
DO=op/op	DO-loop interpretation. <op> is one of: LONG - trip count may be > 131,071 OT - each DO loop is exeuted at least once
DO	Same as DO=OT.
DO=0	Trip count must be <= 131,071 and minimum trip count is 0.
omitted	Same as DO=0.
DS	All C\$ directives are ignored.
DS=0	All C\$ directives are processed.
omitted	Same as DS=0.
E=lfm	Output error list (see EL option) on file <lfm>.
E	Same as E=ERRS.
omitted	Same as E=OUTPUT.
EL=C	List catastrophic errors.
EL=F	Same as EL=C plus fatal errors.
EL=W	Same as EL=F plus warning errors.
EL=T	Same as EL=W plus trivial errors.
EL	Same as EL=F.
omitted	Same as EL=T.

option	action
ET=C	Abort job if catastrophic errors during compilation. The next control statement to be executed is the one following EXIT(S). If no EXIT(S) statement, job ends.
ET=F	Abort job if fatal or higher errors.
ET=W	Abort job if warning or higher errors.
ET=T	Abort job if trivial or higher errors.
ET	Same as ET=F.
ET=O	Continue even if errors in compilation.
omitted	Same as ET=F (at DTNSRDC).
GO	Load and execute object code without a separate LGO. (B=0 and QC cannot be specified)
GO=0	Do not load and execute.
omitted	Same as GO=0.
I=lfm	FORTTRAN source input is on <lfm>.
I	Same as I=COMPILE (see page 16-6,7).
omitted	Same as I=INPUT.
L=lfm	Output lists (BL, LO) are to be put on file <lfm>. (see also E option)
L=0	Listings are suppressed.
L	Same as L=LIST.
omitted	Same as L=OUTPUT.
LO=op/op/...	Listing options (see L parameter). <op> is one of: A - list of variables, common blocks and attributes M - address map O - object code list (use only at request of User Services) R - cross-reference map S - source code list
LO	Same as LO=S/A/R.
LO=0	No A, M, O, R, S information.
omitted	Same as LO=S/A.
MD=T	Machine-dependent usage flagged but considered trivial.
MD=F	Machine-dependent usage flagged and considered fatal.
MD=0	No machine-dependent diagnostics generated.
omitted	Same as MD=0.

option	action
OPT=0	Fast compile (required by DB=ID).
OPT=1	Intermediate optimization.
OPT=2	High optimization, slow compile.
OPT=3	Same as OPT=2 plus potentially unsafe optimization.
OPT	Same as OPT=2.
omitted	Same as OPT=0. (FTN5 OPT=0/1/2/3 approximate FTN4 OPT=0/1/2/UO)
PD=8	Print density. Compile time listings (E, L) are 8 lines per inch.
PD=6	Compile time listings are 6 lines per inch.
PD	Same as PD=8.
omitted	Same as PD=6.
PL=<n>	Specify decimal maximum number of records to be written at execution time on file OUTPUT. The print line limit may be reset at execution time by specifying *PL=<n>, where <n> is the new line limit, at the end of the execute statement. (max: 9 999 999 999) (see page 13-8)
PL	Same as PL=50000.
omitted	Same as PL=5000.
PN	Page numbering of output list is continuous from subprogram to subprogram.
PN=0	Each subprogram starts with page 1.
omitted	Same as PN=0.
PS=n	Page size (number of lines per page in compilation listing. n >= 4)
omitted	Same as PS=80 (if PD=8). Same as PS=60 (if PD=6).
PW=n	Page width (number of characters per line in compilation listing. 50 <= n <= 136)
PW	Same as PW=72.
omitted	Same as PW=72 (if L or E file is connected). Same as PW=136 (all other files).
QC	Quick syntax check. No binary output, no cross reference addresses. Conflicts with B, GO, LO=O/M.
QC=0	No quick syntax check.
omitted	Same as QC=0.

option	action

REW=op/op/...	Rewind option. <op> is one of: B - rewind the binary output file (object code) E - rewind the error file I - rewind the input file L - rewind the output file
REW	Same as REW=I/B.
REW=0	No files rewound.
omitted	same as REW=0.
ROUND=op/op/...	Rounded arithmetic for specified operators. (<op> is one of: A, S, M, D)
ROUND	Same as ROUND=A/S/M (at DTNSRDC).
ROUND=0	Rounded arithmetic will not be used.
omitted	Same as ROUND=A/S/M/D (at DTNSRDC).
SEQ	Sequenced line format.
SEQ=0	Standard FORTRAN format.
omitted	Same as SEQ=0.

*** FTN5 PROGRAM Statement ***

The first statement of a FORTRAN main program must be a PROGRAM statement in the following format (FTN5, 6-2):

PROGRAM name (f1,f2,...,fn) or PROGRAM name

name is the name of the main program (1 to 7 alphanumeric characters, beginning with a letter.

f1,f2,...fn are the names of input/output files used in the main program or one of its subprograms. Each is 1 to 6 alphanumeric characters, beginning with a letter. Up to 50 files may be listed.

Each fj has one of the following forms:

file	- name of a file
file=n	- <n> is the buffer length (octal or decimal) (default: 513)
file=/r	- <r> is the maximum record length for formatted, list-directed or namelist records (default: 150)
file=n/r	- specify both buffer and record lengths
altfile=file	- the two file names are equivalent. <file> must be defined before this specification is given.

If no PROGRAM statement is specified, 'PROGRAM START. (INPUT, OUTPUT)' is assumed by the compiler.

The file name 'TAPEj' is used if READ(j,f), WRITE(j,f), READ(j), WRITE(j), BUFFER IN(j,..., or BUFFER OUT(j,... statements are used. <j> is an integer from 0-999.

If a file is not listed in the PROGRAM statement, a buffer is created when the file is first referenced.

The minimum buffer sizes are:

terminals	- <n> is ignored. Specify 0.
buffered files	- buffers are not used. Specify 0.
disk files	- 64 minimum. Specify > 64 for large records or sequential files.
sequential files	- SI tapes - 128 formatted, 512 unformatted
	S tapes - 512
	L tapes - >= maximum block length
disk	- 64 formatted, 512 unformatted

Record length should be specified for list-directed files.

Equated files should appear after other file definitions, since they cannot be replaced. (page 13-8; FTN5, 11-21)

Examples - PROGRAM CAT (INPUT,OUTPUT,TAPE184,TAPE5=INPUT,TAPE6=OUTPUT)
 PROGRAM TERM (TTY=0,OUTPUT=128,TAPE13,TAPE5=TTY)
 PROGRAM CHECK (OUTPUT=128,TAPE1=128/200,TAPE2=/50)
 PROGRAM BUFR (BUFRFL=0, TAPE46=BUFRFL)

*** Object Program Execution ***

** File Name Replacement **

At execution time, file names may be replaced in the order in which they appear in the PROGRAM statement. For example, if the program statement is

PROGRAM TEST (INPUT, OUTPUT, TAPE1, TAPE2)

The files used at execution time are:

INPUT
OUTPUT
TAPE1
TAPE2

If the program is executed using

LGO(,MYFILE1,MYFILE2)

the files actually used are:

INPUT
OUTPUT
MYFILE1
MYFILE2

Equated files (such as 'TAPE5=INPUT') may not be replaced, and therefore, should be placed at the end of the PROGRAM statement. Note that an equated file and the file to which it is equated are a single file, not two separate files.

Do not use the same file name to replace two or more files for a given program execution.

** Print Limit Specification **

The maximum number of lines written to file OUTPUT is defined at compilation time (page 13-4: PL). It may be overridden at execution time by using

*PL=<new limit>

after any file replacements on the execute statement. For example,

LGO,*PL=10000.
LGO(,MYFILE1,MYFILE2,*PL=600)

**** Post Mortem Dump Parameters ****

Parameters to control post mortem dump output and array subscript limits may be specified at execution time. These parameters follow any file replacements on the execute statement.

***DA=i+j+k+l+m+n+o**

Specify subscript limits. i, j, k, l, m, n, o are integers indicating the maximum subscript values to be printed. They represent the first thru seventh subscripts, respectively, and may be omitted to reduce the amount of output:

If o is omitted, 7-dimensional arrays are not printed.

If n-o are omitted, 6- and 7-dimensional arrays are not printed.

If m-o are omitted, 5- thru 7-dimensional arrays are not printed.

...

If only i is specified, only 1-dimensional arrays are printed.

For example,

LGO(*DA=4+2)

dumps only 1- and 2-dimensional arrays. The maximum subscript of the first dimension will be 4; of the second subscript: 2.

Default: *DA=20+2+1+1+1+1+1.

***OP=<list>** Specifies the format and destination of the dump. <list> is one or more of:

A - All active routines are included in the dump

F - Interactive only. Output is sent to file PMDUMP when file OUTPUT is connected. A message is printed stating this.

T - Interactive only. A condensed form of the output is sent to the terminal if OUTPUT is connected.

*OP=F is the interactive default.

*OP= is the batch default.

For example,

LGO,*OP=AF.

**** User Parameters ****

User parameters may be passed to the executing program by placing them after all file replacement parameters on the execute statement. These parameters are read by the program using subroutine GETPARM (described below).

User parameters have the form:

name
name=value

where name is 1-7 numbers or letters.
value is a string of numbers, letters or asterisk (*). If it contains any other characters, the string must be dollar-delimited (\$<string>\$).

For example,

LGO(IN1,OUT,*PL=7000,A1=47,B=AMDS)
LGO,Z=\$+\$.

**** Accessing User Parameters ****

User parameters on the execute statement are accessed within an FTN5 program using

CALL GETPARM (cname, cvalue, icode)

where cname is a character variable or array element to contain the parameter name.

cvalue is a character variable or array element to receive the parameter value.

icode is an integer return code which will contain one of:

-1 -- end of user parameters
(<cname> and <cvalue> are undefined)
0 -- normal return
+1 -- parameter name only
(<cvalue> contains blanks)

*** FTN4 vs FTN5 ***

The following summarizes the differences between FTN, version 4 (FORTRAN 66 Extended) and FTN, version 5 (FORTRAN 77 Extended).

feature	FTN4	FTN5
comment (column 1)	C, *, \$	C, *
multiple statements	yes, separated by \$	not allowed
column 1-5 of continuation lines	may be non-blank	must be blank
type declarations	TYPE is optional: TYPE REAL A, B -or- REAL A, B	TYPE is not allowed: REAL A, B
number of array dimensions	1-3	1-7
octal constants	nnnnB	O"nnnn"
hexadecimal constants	---	Z"mmm"
octal constant, Hollerith constant, shift or mask prefixed by + or -	sign is typeless operand. result of function is a signed, typeless operand.	sign is arithmetic operator. constant or result of function is a non-arithmetic bit string.
		----- To retain FTN4 meaning, change constants in data statements to unsigned octal constants; in replacement stmts, drop +, change - to .not. and enclose in parens.
Hollerith constants	may exceed word length (10 characters)	may not exceed word length (10 characters)
FORMAT strings	"<string>" *<string>*	'<string>' "<string>"

feature	FTN4	FTN5
alternate form of DATA statement	DATA (nlist=clist),...	not available
non-ANSI forms of constant list in DATA statement	(constant list) rf*(constant list) <rf is repeat factor>	constant list rf(constant list) this avoids conflict with complex constants: (real,imag) rf*(real,imag)
END statement	may be omitted	required
continued END statement	allowed	not allowed
STOP with Hollerith constant	STOP "the end"	STOP 'the end'
unsubscripted array name in replacement statements	treated as first element	not allowed, must have subscript
successive exponentiation	left to right	right to left
complex expression in arithmetic IF	yes, only real part IF (C+CC) 1, 2, 3	no, use REAL function IF (REAL(C+CC)) 1, 2, 3
logical operators in complex expression in logical IF	any operator, only real part used IF (C.GT.CC) GO TO 10	only .EQ. and .NE., use REAL function: IF (REAL(C).GT. REAL(CC)) GO TO 10
2-way arithmetic IF	yes	no
2-way logical IF	yes	no

feature	FTN4	FTN5
DO-loop trip count	one (all loops executed at least once)	zero (may not execute loop) To force one-trip: use FTN5,DO=0T or CS LOOP(0T=1) in program.
DO-loop parameter redefinition	allowed	Allowed but DO-loop limits are calculated once when loop starts.
computed-GOTO expression out of range	fatal	falls thru to next statement
redundant parens in I/O list	allowed	not allowed - may lead to confusion with COMPLEX constants
double precision I/O list items	Allows 2 edit descriptors. (2A10 or 2020)	Must be exactly one edit descriptor.
end-of-file test	EOF function	END= in READ statement (EOF is retained - non-ANSI)
errors in read	IOCHEC	ERR=, IOSTAT= in READ statement
H-format descriptor for input	yes	no - use CHARACTER data
X-format descriptor	Prefixed constant may be omitted. (X means 1X)	Must be prefixed with a non-zero, optionally signed, integer constant.
commas in formats	Optional after H, X, "<string>".	Required after each descriptor.
variable FORMAT in simple variable or non-CHARACTER array element	yes	no - conflicts with assign <format> statement
core-to-core I/O	ENCODE DECODE	WRITE (<array>,... READ (<array>,...

feature	FTN4	FTN5
alternate return from subprogram	SUBROUTINE, RETURNS(ivar) RETURN ivar CALL, RETURNS(200)	SUB (...,*,...) RETURN 1 CALL (...,*200,...)
argument list for entry points to	Not specified in ENTRY statement. assumed same as SUBROUTINE/FUNCTION list.	Specified in ENTRY statement, may differ from SUBROUTINE/FUNCTION list.
using a statement function	Arguments substituted without regard to type or side effects of external function reference.	Each argument is evalu- ated and type converted, if necessary, before substitution. Side effects prohibited.
unique intrinsics	AND/COMPL/OR/XOR	ANINT/BOOL/CHAR/DACOS/ DASIN/DDIM/DFLOAT/DINT/ DNINT/DPROD/DTAN/DININT/ LEN/LOG/LOG10/MAX/MIN/ NINT
modified intrinsics	---	Several intrinsics have been replaced by zero- argument intrinsics.
sense lites	SLITE/SLITET subprograms	none (use logical variables)
debug facility	C\$ DEBUG	none (see CID)
compiler directives	C/ LIST,ALL C/ LIST,NONE	C\$ LIST(ALL) C\$ LIST(ALL=0) many more available

*** New Features of FTN5 ***

1. ASSIGN may assign a format number.
2. CHARACTER data type.
3. IF-THEN-ELSE construct.
4. INTRINSIC statement allows an intrinsic to be passed as an argument.
5. I/O statements: OPEN, INQUIRE, CLOSE;
END=, ERR=, IOSTAT= in READ/WRITE statements.
6. Format descriptors:
 - S/SP/SS - output sign control
 - BN/BZ - input blank interpretation
 - : - if no more data, terminate format
7. PARAMETER statement to give a symbolic name to a constant.
8. SAVE specification statement to save values in a subprogram from one invocation to another.

*** FORTRAN, Version 5, Considerations ***

1. Using Post Mortem Dump ('FTN5,DB=PMD' or 'FTN5,DB') suppresses all load map directives. A block map is written into file ZZZZZMP, which can be printed by rewinding and copying it, or by routing it to a printer. (FTN5, 10-3)
2. If functions CHAR, ICHAR and/or INDEX are used, a collating sequence must be defined. (page 13-2: CS)
3. A program which calls Sort/Merge cannot specify fixed-length argument lists ('FTN5,ARG=FIXED'). (FTN5, 11-2, 8-5)
4. Processing with CHARACTER data is slow on CDC computers.
5. When using BUFFER IN/BUFFER OUT, leave an extra word after the array to provide a place for the tape error status word. Failure to do so may cause the contents of the "good" word after the array to be erased.
6. FTN4 subprograms using non-standard returns cannot be called from FTN5, and vice versa.
7. FTN4 subprograms which do no I/O can be called by FTN5, and vice versa.
8. FTN5 programs may call FTN4 subprograms which do standard I/O. Programs with intermixed FTN4/5 I/O must not specify STATIC. They should specify the global library set:
LIBRARY(FTN5LIB,FORTRAN)
9. CHARACTER data cannot be passed from FTN5 to FTN4.
10. It is more efficient to declare CHARACTER dummy arguments with a length of (*) instead of a constant length.
11. It is more efficient to dimension dummy arguments as (*) instead of (1). It may also reduce the number of compiler warning messages.
12. A format defined in a READ or WRITE statement is not validated by the compiler. To catch errors at compile time, instead of execute time, use a FORMAT statement.

13. The ANSI standard (section 12.1.3) states that "[a]n endfile record may occur only as the last record of a file." Thus, FORTRAN 77 does not support CDC files which have embedded EOF's. To read such a file, use the END= in the READ statement. At the branch label, use the CDC EOF function, which resets the endfile indicator and allows you to read the next file in the file. If this method is used, there is no way to know when end-of-information has been reached. Therefore, you must know how many files there are and count them in the program, or you must have a data record to indicate the real end of the file. For example,

```
...  
100 CONTINUE  
...  
    READ (1, 1, END=500) A, B, C  
...  
500 CONTINUE  
    E = EOF (1)  
    NFILE = NFILE + 1  
...  
    IF (NFILE .LT. MAXFYL) GO TO 100  
...
```

Any program processing multi-file files should be avoided because this is non-standard and, therefore, non-portable.

*** FORTRAN, Version 5, Hints ***

1. The built-in function INDEX locates the first occurrence of one string within another string. A not so obvious use is to see if a character belongs to a set of characters. Thus,

```
CHARACTER * 1 CH
IF (INDEX('0123456789', CH) .NE. 0) THEN
...
```

tests whether 'CH' is a digit.

2. The built-in function CSOWN allows you to redefine the collating sequence. Any character not specified will collate equal to the lowest (first) character in the list. One use of this might be to scan a line for English words:

```
CHARACTER LINE * 80
CALL CSOWN (' ABCDEFGHIJKLMNOPQRSTUVWXYZ')
...
C    test for delimiter (non-letter)
    IF (LINE(I:I) .NE. ' ') THEN
...
```

(CS=USER must be specified in the FTN5 statement.)

3. A trick to check for undeclared or misspelled variables: compile using 'IMPLICIT LOGICAL (A-Z)'. Virtually all undeclared or misspelled variables will cause an error.

*** FTN4 to FTN5 Conversion Aid ***

A conversion aid program, F45, is available to assist in converting programs from FORTRAN Extended, Version 4 (ANSI FORTRAN 66), to FORTRAN, Version 5 (ANSI FORTRAN 77). The detailed description of its operation is found in "FORTRAN Extended Version 4 to FORTRAN Version 5 Conversion Aid Program Version 1 Reference Manual" (CDC publication no. 60483000).

Note that if the first card of the source deck looks like an UPDATE COMPILE file (an ID field followed by a space followed by a sequence number all starting after column 72 and ending after column 80), F45 will produce only UPDATE directives. These directives will probably be unusable.

A source deck which is not considered an UPDATE COMPILE file may have any information after column 72 changed (comment lines are retained; unmodified lines are truncated after column 80; modified lines are truncated after column 72).

Field length required: 75000 CM words.

** Examples **

- 1) Convert source deck on permanent file and catalog the converted program:

```
jobname,CM75000,....          name/code
CHARGE,....
ATTACH,MY4,....               <-- FTN4 source program
REQUEST,MY5,*PF.
F45,I=MY4,P=MY5,PO=F,LO.
CATALOG,MY5,....             <-- FTN5 source program
<eoi>
```

- 2) Like 1, but using procedure F45IT (so you don't have to remember the F45 parameters):

```
jobname,CM75000,....          name/code
CHARGE,....
ATTACH,MY4,....               <-- FTN4 source program
REQUEST,MY5,*PF.
BEGIN,F45IT,,MY4,MY5,OUTPUT.
CATALOG,MY5,....             <-- FTN5 source program
<eoi>
```

- 3) Convert source deck from an UPDATE library and catalog the new library with the converted program:

```
jobname,CM75000,....           name/code
CHARGE,....
ATTACH,OLDPL,....             <-- UPDATE library with FTN4 program
UPDATE.                       <-- extract the FTN4 program
F45,I,P=MY5,L0.
REQUEST,NEWPL,*PF.
UPDATE,I=MY5,N.
CATALOG,NEWPL,....           <-- UPDATE library with FTN5 program
<eor>
*COMPILE MY4
<eoi>
```

- 4) Convert source deck on cards, call FTN5 to compile, execute:

```
jobname,CM75000,....           name / code
CHARGE,....
F45,PO=F,P=NEWIN,L0.          <-- convert and get full source output
FTN5,I=NEWIN,REW=I.
LGO.
<eor>
    (FTN4 source program)
<eor>
    (data for program, if any)
<eoi>
```

- 5) The converted program may need to be run on a non-CDC computer, so flag all machine-dependent statements:

```
...
F45,MD,....
...
```

*** Error Messages ***

** FORTRAN Compilation Errors **

FTN5 prints the diagnostic messages immediately after the erroneous source line. Usually, errors will be trivial, warning or fatal.

For example,

TRIVIAL	*	NO PATH TO THIS STATEMENT
WARNING	*	NUMBER OF ARGUMENTS IN REFERENCE TO _ABCD IS NOT CONSISTENT
FATAL	*	TOO FEW RIGHT PAREN OR PREVIOUS SYNTAX ERROR -- SCAN STOPPED AT _EOS-
FATAL	*	ILLEGAL USE OF OPERATOR / OPERAND -- *
FATAL	*	STATEMENT LABEL .998 REFERENCED BUT NOT DEFINED
FATAL	*	LEFT SIDE OF EQUAL SIGN IS ILLEGAL

Fatal compilation errors will abort the job unless A=0 option was used, in which case an attempt to load the program will proceed until it encounters an uncompiled routine, flagged by an illegal name. The following message then appears in the dayfile:

FATAL LOADER ERROR -
ATTEMPT TO LOAD SUPPRESSED BINARY

The following messages mean the field length (CM) must be increased:

TABLE OVERFLOW -- INCREASE FIELD LENGTH AND RERUN
TABLE OVERFLOW IN routine-name
FTN5 NEEDS AT LEAST <cm> CM FIELD LENGTH

** FORTRAN Object Time Errors **

The FORTRAN math library and I/O routines detect many object-time errors. Some are informative and produce a message in the output listing only. Others are fatal, producing the message "FTN - FATAL ERROR nn" in the dayfile and a more detailed message in file OUTPUT. The operating system detects certain other errors, such as arithmetic mode errors. FTN5 control statement option DB=ER (default with OPT=0) reprieves these and interprets their meaning and location in the dayfile, such as "INFINITE VALUE IN XYZ NEAR LINE n. Fatal errors cause all output file buffers to be emptied, but not other file buffers.

Subroutines LIMERR and NUMERR may be utilized to allow the program to accept certain input data errors. Subroutine RECOVER with suitable auxiliary routine may be used to gain temporary control after otherwise fatal system errors. Subroutine SYSTEM may be used to change the treatment of errors. (see page 13-22)

*** FORTRAN-Callable Utilities ***

GETPARM Read the next user parameter on the execute statement.
 (see page 13-10; FTN5, 7-14)

LIMERR 'CALL LIMERR (max)' allows the user to encounter <max>
 errors in FORTRAN formatted input data before fatal
 termination (default is 0). LIMERR resets the NUMERR
 error count to 0. (see FTN5, 7-29; FTN4, 8-20)

NUMERR 'NUMERR ()' is an integer function which returns the
 number of errors since the last call to LIMERR. (see
 FTN5, 7-29) The FTN4 version requires a dummy argument
 'NUMERR (n)'. (see FTN4, 8-20)

RECOVR should be used to gain temporary control after an abort
 condition to close files which an interactive Intercom job
 might otherwise leave open, and to close output buffers
 for files needed for a later job, such as Calcomp (see
 FTN5, 7-16; FTN4, 8-12; NOSBE, 7-33). For example,

EXTERNAL MYEND
CALL RECOVR (MYEND, 0"77", 0)

where you supply

SUBROUTINE MYEND (EX, IFLAG, RA)
DIMENSION EX(17), RA(1)
C close Calcomp file and end file alternate output
CALL PLOT (0, 0, 999)
END FILE 12
STOP
END

SYSTEM 'CALL SYSTEM (ernum, msg)' issues an execution-time error
 message. <ernum> is a decimal integer from 0 thru 9999.
 Error numbers 51 (nonfatal) and 52 (fatal) are reserved to
 the user. Other numbers retain the severity associated
 with them. If <ernum> exceeds the highest assigned error
 number (FTN5 and FTN4, Appendix B), 52 is assigned.
 <msg> is the error message. This is a CHARACTER constant
 with the first character treated as carriage control.

SPY

will help locate areas in a program where the code is inefficient. SPY will record results on file DOSSIER showing how much relative time was spent in each <binw> words from <low> to <high>.

To turn on SPY:

```
CALL SPYONF (low, high, name, binw)
  low - start (relative to RA) of area to be analyzed
  high - end (relative to RA) of area to be analyzed
  name - label for this invocation of SPY
         left-justified, 1-8 characters (e.g., "ABC")
  binw - width of SPY analysis. May be 0"100", 0"40",
        0"20", 0"10" (or 64, 32, 16, 8).
        In most instances, 0"100" (64) is sufficient.
```

To turn off SPY:

```
CALL SPYOFF
```

Before the program terminates, it must "CALL SPYOFF".
Subroutine RECOVER may be needed (see page 13-22).

Example:

```
...
CALL SPYONF (0"111", 0"40000", "MYPROG", 0"100")
...
CALL SPYOFF
...
```

(See page 5-8: PRNTSPY, for directions on printing the resulting SPY analysis. See also CONV, chapter 11.)

For additional FORTRAN-callable control statement routines, such as

```
LF      (REQUEST, ROUTE, RETURN, etc.)
PF      (ATTACH, CATALOG, etc.)
SKPFIL
```

See page 18-7: NSRDC; CLIB/N.

*** Sample FORTRAN Deck Setups ***

** Compile and Load Only **

```
jobname,....          name/code
CHARGE,....
FTN5.                  <-- see options on page 13-2
LOAD(LGO)              <-- checks for missing subroutines
NOGO.                  <-- causes load map to be generated
<eor>
    (all FORTRAN source decks)
<eoi>
```

** Compile and Execute (Using PMD for Debugging) **

```
jobname,....          name/code
CHARGE,....
FTN5,DB,GO.            <-- compile, load and execute
EXIT,U.
REWIND,ZZZZZMP.
COPYE,ZZZZZMP,OUTPUT.  <-- print load map
<eor>
    (all FORTRAN source decks)
<eor>
    (data deck)
<eoi>
```

** Compile and Execute, Catalog Binary Program **

```
jobname,....          name/code
CHARGE,....
REQUEST,BINARY,*PF.    <-- get space for permanent file
FTN5(B=BINARY,OPT=1)   <-- object code to file BINARY
MAP(ON)                <-- full map for documentation
BINARY.                <-- load from file and execute
CATALOG,BINARY,PROGBIN,ID=xxxx,XR=RDONLY.  <-- make file permanent
<eor>
    (all FORTRAN source decks)
<eor>
    (data deck)
<eoi>
```

** Compile, Load and Execute (with Library Routines) **

```
jobname,CM100000,T100.                name/code
CHARGE,....
FTN5(OPT=2)                            <-- see options above
ATTACH,IMSL.                           <-- get IMSL library
LDSET,LIB=IMSL.                        <-- make IMSL available to the loader
LGO.                                   <-- load and execute
<eor>
    (all FORTRAN source decks)
<eor>
    (data deck)
<eoi>
```

** Execute Only (Binary Program on Permanent File) **

```
jobname,....                name/code
CHARGE,....
ATTACH(OBJECT,PROGBIN,ID=xxxx,MR=1) <-- retrieve permanent file
OBJECT.                          <-- load file and execute
<eor> (not needed if no data)
    (data deck)
<eoi>
```

** Compile and Execute (Tape Used) **
(Dump, if Execution Errors)

This job needs two tapes: TAPE27, a phase-encoded binary output tape and TAPE8, a hyper-density, formatted slot tape. Binary input data on disk file TAPE16.

```
jobname,MT1,PE1,....          name/code
CHARGE,....
FTN5.
ATTACH,TAPE16,ID=xxxx.
VSN,TAPE8=SLOT22=xxxx02.
LABEL,TAPE27,l=xxxxDATA01,D=PE,W,VSN=CA8888,RING.
LABEL,TAPE8,L=xxxxDATA02,D=HY,R,RING.
LGO.
RETURN,TAPE16,TAPE27,TAPE8.
EXIT.                          <-- following executed only after error
DMP,46000.                     <-- dump first 46000 of field length
RETURN,TAPE16,TAPE27,TAPE8.
<eor>
```

PROGRAM LINE

```
...
READ (16) (A(I),I=1,32)          ** - see below
...
WRITE (8,5) (B(I),I=1,32)       ** - see below
WRITE (27) (B(I),I=1,32)       ** - see below
...
END
<eor> (not needed if no data)
      (data deck)
<eoi>
```

** - Program would be more efficient, assuming 'DIMENSION A(32), B(32)' is used, if the I/O lists used just the array name (e.g., 'READ (16) A').

*** FORTRAN Notes ***

1. Compile the program with default 'OPT=0' until it is error-free. Production programs should be compiled with 'OPT=1' or 'OPT=2' (FTN5 only).
2. Division by zero does not yield a zero as some other computers do. Using the result will cause a halt in execution on the CYBER 750. It will cause a mode 2 or 4 error on the CYBER 176. The built-in function 'LEGVAR(v)' will test for infinite and indefinite quantities. The function will return -1 for an indefinite, +1 for an infinite, and 0 for normal. User may also test for the divisor equal to zero and, if so, set the result to zero, if desired.
3. If more than one BLOCK DATA subprogram is used, each must have a name (FTN5, 6-3; FTN4, 7-5).
4. Specification statements (DIMENSION, EQUIVALENCE, COMMON and TYPE) must appear before DATA, arithmetic statement function definition and NAMELIST statements, which in turn must come before any executable statement.
5. Double precision constants are defined by the D exponent. Due to the 48-bit coefficient in CDC single precision, few operations require double precision.
6. Mixed-mode expressions (e.g., $A=B+2$) are allowed and no error diagnostic will result. However, this is not standard FORTRAN and should be used with care. For example, $A^{**}(1/3)$ equals 1.
7. On the CDC CYBER, there is no way to explicitly define the length of a non-character variable. (e.g., 'REAL*8 MASS' must be changed to 'REAL MASS' or 'DOUBLE PRECISION MASS'.)
8. A variable name within a subroutine cannot be the same as the subroutine name.
9. Dummy arguments of a subroutine may not appear in a COMMON, DATA or EQUIVALENCE statement in the subroutine.
10. All arguments to subroutine or function subprograms are transferred by location (name) on the CDC CYBER (except for library routines, which will have values transmitted by value unless the external function name appears in an EXTERNAL statement in the calling program). Slashes used in IBM subroutines to pass undimensioned arguments by name must be removed.

11. Missing subprograms do not inhibit execution of a simple or overlay program. If a missing routine is actually called, the run will abort with a dayfile message 'ERROR MODE 1, LOCATION 4xxxxx' (see page 15-3). Unsatisfied references will be listed in the loader map.

12. Variables occurring in subscripts of three or more dimensions cause slower execution, especially in double precision or complex arrays.

13. Attempt to print numbers invalid for the format field will cause one of the following results:

- * in the entire field if the number is too large for the field.
- R if value is infinite (range error)
- I if value is indefinite

14. Random access to data on disk may be accomplished on CDC by callable system routines (e.g., READMS). (FTN5, 7-22; FTN4, 8-28) Since such random data is not buffered ahead, it is slow. 'OPENMS' is called to open the random file and set up the index pointers. The user must not zero the index array after OPENMS. 'CLOSMS' may be called to close the file, but it is not required. Block as well as index length should be a multiple of 64 words for speed.

15. Current memory buffers for written files not equated to OUTPUT are not flushed (emptied) to the file if a job aborts. Thus, lines of data may be lost. RECOVER (page 13-22; FTN4, 8-12; NOSBE, 7-33) may be used.

16. If the primary entry name of a multiple entry system library routine is replaced by a routine of the same name in user's program, all secondary entries, if referenced, must be replaced also. Unsatisfied externals will result otherwise.

17. Subroutine MOVLEV moves arrays between main memory and Extended Core Storage and must not be used at DTNSRDC. Instead, use subroutine MOVEIT or MOVECM in library NSRDC (see page 18-7).

18. Comments should not occur following the last END statement. The compiler will print the non-fatal dayfile message 'NULL PROGRAM IGNORED AFTER x...x', where x...x is the name of the last routine.

19. Binary (unformatted) files may not set the buffer size on the PROGRAM statement below 512 words unless special file type. If it is necessary to use smaller buffers, the following control statements will be required:

```
FILE(TAPEnn,BT=C,RT=S)
XEQ,LDSET=FILES=TAPEnn,LOAD=LGO
```

20. FORTRAN READs and WRITEs utilize Record Manager to process all I/O requests. The user may bypass FORTRAN I/O processing by calling Record Manager routines directly using 'CALL GET' and 'CALL PUT'. In addition, there are routines for file definition, open, close, rewind, forward space, backspace. These routines allow the user to process files which are not in one of the three standard FORTRAN types. (FTN5, 8-1; FTN4, 8-39; BAMUG; AAMUG)

21. A program calling Sort/Merge may not dynamically allocate memory (by over-indexing blank common or by using routines such as REDUCE).

22. If object file (LGO) has been cataloged, even though some FORTRAN subprograms did not compile correctly, COPYL (page 12-2) may be used to replace the bad compilations from a new object subfile. If the bad routine was a main program in an overlay job, the OVERLAY line must be inserted by COPYN (page 12-3). When an OVERLAY line is being copied, begin in column 1 as any other NOS/BE control statement.

23. When using reprieve ('FTN5,DB=ER' or 'FTN4,ER'), object code will include an extra instruction for each statement for use by reprieve, if job aborts. ER is default with OPT=0.

24. When a program is explicitly compiled under Intercom you must first 'REWIND,lfn,LGO.', where <lfn> is the file with the source program. To retrieve compiler error messages, use 'ERRORS,FTN'. (see CLIB/P: RUNFTN)

Another method is to 'CONNECT,OUTPUT' and use 'L=0' in the FORTRAN control statement. In this instance, error messages and the statements in error are listed at the terminal and there will be no listing which can be routed to a printer.

25. When using BUFFER IN/BUFFER OUT, leave an extra word after the array to provide a place for the tape error status word. Failure to do so may cause the contents of the "good" word after the array to be erased.

*** FTN4 Control Statement Parameters ***
 (see FTN4, chapter 10)

Examples FTN4. defaults to (A,B=LGO,BL=0,EL=I,ER,I=INPUT,L=OUTPUT,
 OPT=0,P=0,PD=6,PL=5000,R=1,ROUND=*/,SL,T)
 FTN4(GO) compile and execute
 FTN4,I=MYFILE,ROUND,OPT=1.
 FTN4(B=PUNCHB,EL=A)

Note: FTN4 and FTN are synonyms.

option	action
A	Abort job if errors occur during compilation. Next control statement to be executed is the one following EXIT(S). If there is no EXIT(S) statement, the job ends.
A=0	Continue even if errors in compilation (forced if D used).
omitted	Same as A (at DTNSRDC).
B=PUNCHB	Produce punched binary decks of all routines; no BCD sequencing in the last columns of the cards.
B=lfm	Put binary images on file <lfm>.
B=0	No binary output.
omitted	Same as B=LGO.
BL	Burstable list. Each major section of compilation listing starts on a new page.
BL=0	Compact list. New page for first page only.
omitted	Same as BL=0.
D=lfm	Debug mode (forces A=0, OPT=0, T; conflicts with TS) Special comment lines (C in column 1 and \$ in column 2) may be used to aid in debugging FTN programs when the D option is specified. These features include array bound checking, dumping variables when changed by a replacement statement, and inter- and intra-program tracing. <lfm> is the file containing the C\$ directives.
D	Same as D=INPUT.
D=0	C\$ debug statements ignored.
omitted	Same as D=0. (see FTN4, chapter 9; DEBUG)

option	action
DB=ID	CYBER interactive debug. Must be specified if CID is to be used and the DEBUG control statement has not been included. When specified, the object code contains a line number table and a symbol table. (level 508 on)
DB	Same as DB=ID.
DB=0	No debug tables are generated. If CID was turned on by a DEBUG control statement, DB=0 turns it off for the compilation.
omitted	Same as DB=0.
EL=F	List fatal diagnostics.
EL=W	OPT: same as EL=F. (level 508 on) TS : same as EL=F plus warning diagnostics.
EL=N	OPT: same as EL=F. (level 508 on) TS : same as EL=F plus notes.
EL=I	OPT: same as EL=F plus informative diagnostics. TS : same as EL=N.
EL=A	Same as EL=T plus diagnostics indicating non-ANSI usage.
omitted	Same as EL=I.
ER	Generate code for object-time reprieve of errors.
ER=0	Do not generate reprieve code.
omitted	Same as ER (TS, OPT=0). Same as ER=0 (OPT=1,2).
GO	Load and execute object code without a separate 'LGO.'. (conflicts with Q, E, B=0)
GO=0	Do not load and execute.
omitted	Same as GO=0.
I=lfm	FORTTRAN source input is on <lfm>.
I	Same as I=COMPILE. (see page 16-6,7)
omitted	Same as I=INPUT.
L=lfm	Output lists (BL, EL, OL, R, SL) are to be put on file <lfm>.
L=0	All listings are suppressed, except fatal diagnostics and the statements causing them.
omitted	Same as L=OUTPUT
OL	List object code (use only when requested by 1892) (conflicts with Q, E).
OL=0	Do not list object code.
omitted	Same as OL=0.

AD-A150 162

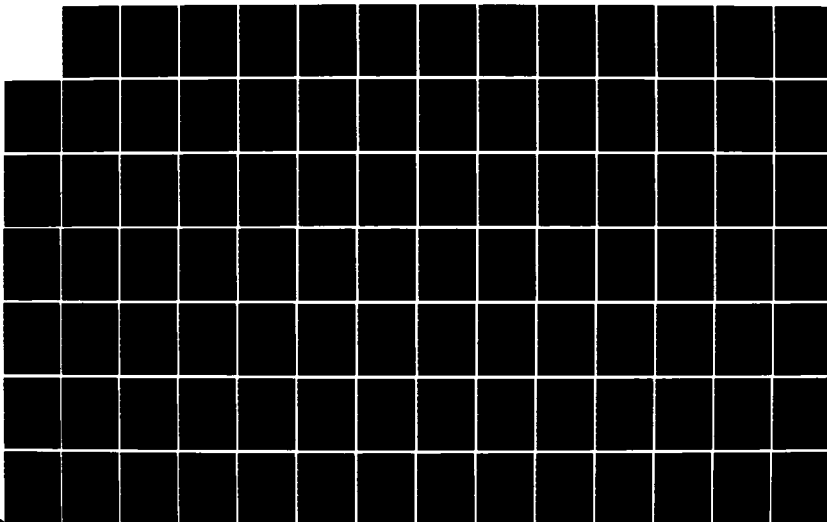
COMPUTER CENTER CDC REFERENCE MANUAL(U) DAVID W TAYLOR
NAVAL SHIP RESEARCH AND DEVELOPMENT CENTER BET..
D V SOMMER ET AL. SEP 84 DTNSRDC/CMLD-84-10

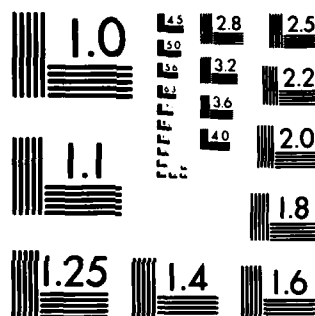
3/5

UNCLASSIFIED

F/G 9/2

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

option	action
OPT=0	Fast compile.
OPT=1	Standard optimization (use for production).
OPT=2	Fast object code (slower compile). (not recommended)
OPT	Same as OPT=2.
omitted	Same as OPT=0 (at DTNSRDC). (OPT conflicts with TS and SEQ) (also see parameters Q and TS; page 13-27: item 34; page 13-36)
P	Page numbering of output list is continuous from subprogram to subprogram.
P=0	Each subprogram starts with page 1.
omitted	Same as P=0.
PD=8	Compile time listings are 8 lines per inch (print density)
PD=6	Compile time listings are 6 lines per inch.
PD	Same as PD=8.
omitted	Same as PD=6. (do not change if program contains embedded COMPASS code)
PL=n	Specify decimal maximum number of unit records to be written at execution time on the file OUTPUT. The print line limit may be reset at execution time by specifying PL=<n>, where <n> is the new line limit, anywhere on the LGO statement. (max: 9 999 999 999)
omitted	Same as PL=5000. (see page 8-4: name)
PMD	Post Mortem Dump will be used. Symbol tables are written to separate files for use by PMD. (level 508 on)
PMD=0	No symbol table files are generated.
omitted	Same as PMD=0.
PS=n	Page size (number of lines per page in compilation listing. n >= 4)
omitted	Same as PS=80 (if PD=8) Same as PS=60 (if PD=6)
PW=n	Page width (number of characters per line in compilation listing. 50 <= n <= 136) Valid only with TS mode.
PW	Same as PW=72.
omitted	Same as PW=72 (if L file is connected). Same as PW=136 (if L file is not connected).

option	action
Q	Fast compile. no object code. no addresses in reference map. No execution possible. (conflicts with B, C, GO, OL, TS, E)
Q=0	Normal compilation.
omitted	Same as Q=0.
R=0	No reference map.
R=1	Short map (symbols, addresses, type, DO loops).
R=2	Long map (short map plus line number references).
R=3	Long map plus common block members and equivalence classes
R	Same as R=2.
omitted	Same as R=1.
ROUND=ops	Rounded arithmetic for specified operators (any or all of +, -, *, /).
ROUND=0	Rounded arithmetic will not be used.
ROUND	Same as ROUND=+*/*.
omitted	Same as ROUND=*/* (default at DTNSRDC)
SEQ	Sequenced line format (forces TS).
SEQ=0	Standard FORTRAN format. (default)
omitted	Same as SEQ=0.
STATIC	Static loading. Inhibits dynamic memory management at execution time by CRM. Required for programs which dynamically extend blank common. (level 508 on)
STATIC=0	CRM uses dynamic memory management at execution time.
omitted	Same as STATIC=0.
T	Turn on error traceback in library routines.
T=0	Turn off error traceback. (e.g., square root of negative number will be flagged as an error, but will not show where the error occurred.)
omitted	Same as T=0 (OPT=1,2). Same as T (OPT=0, D).
TS	Time-sharing mode. OPT ignored if TS specified.
omitted	Same as OPT=0. (at DTNSRDC)

*** FTN4 PROGRAM Statement ***

The first statement of a FORTRAN main program must be a PROGRAM statement in the following format (FTN4, 7-2):

PROGRAM name (f1,f2,...,fn)

where 'name' is the name of the main program (from 1 to 7 alphanumeric characters, beginning with a letter). The parameters 'fj' are the names of all input/output files required by the main program and all subprograms. Each is 1 to 6 alphanumeric characters, beginning with a letter. No more than 50 files may be specified.

1. File OUTPUT should always be specified for possible error messages.
2. For the file name 'TAPEj', <j> is an integer from 1-99. If <j> is a variable, there must be a file name 'TAPEj' for each value <j> may assume.

*** FORTRAN Version 4 Notes ***

1. The file 'DEBUG=OUTPUT' should appear in the PROGRAM statement if the FORTRAN debug option might be invoked and the user desires interspersed debug output. Otherwise the debug file will be routed to the printer by the system at end-of-job.
2. If a file may reside on tape, increasing the buffer size to 1024 (double buffering) should improve running time.
3. Formatted or namelist input and output has a default limit of 150 characters per unit record. Printer output greater than 137 characters is printed on 2 lines. The size may be altered in the PROGRAM statement (e.g., 'TAPE1=/400,TAPE2=/121') (FTN4, 7-2); or direct calls to Record Manager may be used (FTN4, 8-39); or 'BUFFER IN', 'BUFFER OUT', 'DECODE' and 'ENCODE' statements may be used, along with 'UNIT' and 'LENGTH' functions (FTN4, 8-23), to perform formatted I/O for records in excess of the limitations specified above.
4. ENCODE/DECODE buffer size is restricted to 150 characters or less. multiple ENCODE/DECODE statements are required for buffers greater than 150 characters. This method is slow.
5. FTN4,OPT=2 may cause incorrect optimization for some programs which use EQUIVALENCE, variable length subprogram calls, or direct Record Manager calls (GET/PUT) or some other "unknown" constructs and thus should be avoided.
6. 'WRITMS' has two additional optional parameters. The NOS/BE default is not rewrite in place. To use this more efficient rewrite, the user must 'CALL WRITMS (u,f,n,k,-1)'. If the -1 parameter is omitted, all rewrites will go at the end of the file. (FTN4, 8-30))
7. FTN4 level 508 (or later) subprograms doing any I/O may not be mixed with level 461 (or earlier) main programs and vice versa.

*** Time-sharing FORTRAN ***

Time-sharing mode FORTRAN (FTN4,TS, see page 13-33) is a one-pass compiler. Because fewer disk accesses are required, compilation is faster than in optimizing mode (OPT=0, 1, 2). Object code is not highly optimized and executes about as fast as OPT=0 code. FTN4,TS requires CM40000 minimum.

In addition, TS FORTRAN will accept some keyword misspellings and punctuation errors. Warning diagnostics will be issued since such errors may be fatal in optimizing mode.

Diagnostic messages are listed after the line where the error was detected. (Optimizing FORTRAN groups them after the complete program listing.)

Both the program listing and the cross reference maps have formats different from those of optimizing FORTRAN. (see FTN4, 11-6, 13-15)

In addition to the standard FORTRAN format (statement label in 1-5, continuation in 6, statement in 7-72), TS FORTRAN also recognize sequenced line mode (sequence number, followed by blank, + or other character, followed by (optional) statement label, followed by statement (thru column 80)). For example,

```
20 PROGRAM TEST (OUTPUT)
40C  compute area
60 REAL A(100), B(100),
80+ C(200)
100 10 CALL SUB (A, B, C, 100)
120 STOP
140 END
```

To indicate sequenced line mode, use SEQ in the FTN4 statement.

See CLIB/P: RUNTS.

*** Minnesota FORTRAN (MNF) ***

The Minnesota FORTRAN compiler (MNF), written at the University of Minnesota, is available on the DTNSRDC CDC CYBER computers.

The MNF compiler may accessed as follows:

```
jobname,CM54000,....      name / code
CHARGE,....
ATTACH,MNFTN.
ATTACH,PROD,461PRODUCTS,ID=CSYS.
LIBRARY,MNFTN,PROD.
MNF,<options>.
<eor>
      (FORTRAN source program)
<eor>
      (data cards, if any)
<eoi>
```

Some of the <options> are:

I means I=COMPILE.

I=lfm Read source statements from file <lfm>.

B Compile only, do not load-and-go. Put binary output on LGO.

B=lfm Compile only, do not load-and-go. Put binary output on <lfm>.

L=lfm Put program and cross reference listings on file <lfm>.

T Turn on 6 object-time tracing features (including subscript
 bound checking) with no change to source code.

P=nnnn Maximum (decimal) number of lines on file OUTPUT at
 execution time (includes any trace output). (default: 5000)

Examples: MNF(I,T,P=1500)

MNF. (same as MNF,I=INPUT,L=OUTPUT,P=5000.)
Binary object code is put on LGO. Program is loaded
with undefined memory set to negative indefinites with
addresses, and executed. Note that no 'LGO.'
statement is needed.

The MNF compiler performs some consistency checks among related programs and subprograms. Therefore, it is useful to compile related routines together whenever possible.

MNF has the same execution-time errors as FTN (improper arguments for math routines, etc.), but, unlike FTN4, they are all fatal.

Compatibility with FTN4:

MNF is a different compiler from CDC FORTRAN, version 4, Extended (FTN4), but it compiles basically the same language. The CDC FTN4 manual may be used, in general, as the reference manual for things other than the control statement options. A program compiled by MNF may call subprograms compiled by FTN4, for example, the math subroutine libraries such as IMSL.

Advantages:

MNF compiles faster than FTN4 OPT=0, has more and better diagnostics better compilation and cross reference information, more powerful debugging and tracing capability. It anticipates some features of the ANSI FORTRAN 77 standard, such as PARAMETER and IF-THEN-ELSE. It allows END= and ERR= in READ statements.

Disadvantages:

Execution time of compiled programs is slower than FTN4 OPT=1 or 2. Therefore, after a program is debugged, it should be recompiled at OPT=1 for normal usage.

It does not allow multiple-entry and multiple-return subprograms.

It requires the use of an obsolete system product set.

A document (10 pages) describing MNF may be obtained by:

BEGIN,DOCGET,,OTHER,,MNF,OUTPUT,MSACCES=<pw>.

*** Rational FORTRAN (RATFOR) ***

Rational FORTRAN is a programming language that has the structure forming statements that allow "top down" and "GO TO-less" programming. R.TFOR is used as a software engineering tool to aid in producing and maintaining complicated computer programs. RATFOR is a pre-compiler (FTN4 must be executed after RATFOR). RATFOR is closer to the ANSI FORTRAN 77 standard (FTN5) than to FTN4. FTN5 is preferred since it is the new standard (RATFOR generated an FTN4 program), and it compiles directly into machine language (a RATFOR run still requires an FTN4 compiler call).

A document (12 pages) describing RATFOR may be obtained by

BEGIN,DOCGET,,OTHER,,RATFOR,OUTPUT,MSACCES=<pw>.

***** COBOL *****

NOS/BE at DTNSRDC has two COBOL compilers:

COBOL5 - COBOL, version 5 (ANSI 1974 standard)

COBOL - COBOL, version 4 (ANSI 1968 standard)

For details of the COBOL5 language, see page i: COBOL5 and pages 14-2 ff in this manual.

For details of the COBOL 4 language, see page i: COBOL4 and pages 14-11 ff in this manual.

References: COBOL 5 Reference Manual CDC 60497100
COBOL 4 to COBOL 5 Conversion Aid CDC 19265021
Federal Information Processing Standards, Dec 1975,
publication number 21-1 (gives information regarding
Federal guidelines for using ANSI 74 COBOL compiler)
A COBOL Version 5 User's Guide CDC 60497200

*** COBOL5 ***

COBOL5, which is CDC's implementation of the 1974 ANSI COBOL standard, is available on all the CDC computers. COBOL5 is in addition to, but does not replace, COBOL 4. Almost all COBOL 4 programs will need some conversion to compile in COBOL5. Compilation in COBOL5 is considerably slower. Execution may be much faster, particularly where sorting is involved.

Control statements for NOS/BE are:

```
... at least CM65000 on job card
...
COBOL5,<options>.
LGO.
<eor>
      (COBOL 74 source program)
<eoi>
```

CAUTION!

When using any version of COBOL5 from a user library rather than the system version, the <lfm> for the library may not be COBOL5 (or any other entry point in the library). Usually, it is necessary to attach the library even when executing from an absolute COBOL5 object file. Use the same <lfm> for the library when compiling and executing, e.g.,

```
ATTACH,MYPROG,ID=xxxx.      <-- get absolutized program
ATTACH,COB5LIB1vl1,ID=CSYS. <-- lvl is the compiler level
MYPROG.                    <-- execute program
```

** COBOL5 Parameters **

The parameters available on the COBOL5 control statement do not all correspond to those used by COBOL 4. In the discussion below, some of the parameters have two defaults. They will be shown as (default: default1/default2) where default1 is when the keyword is omitted and default2 is when the keyword is given alone. For example, the defaults for the 'I' option are input/compile. This means that when the 'I' parameter is omitted (COBOL5,...), the default is INPUT and when the 'I' parameter is given alone (COBOL5,I,...), the default is COMPILE. Some of the more frequently used parameters are:

option	action
B=lfm	File to contain relocatable binary object program. (defaults: LGO/BIN) Use B=0 to suppress binary output.
I=lfm	File containing source program. (defaults: INPUT/COMPILE)
L=lfm	File to contain compilation listings. (defaults: OUTPUT/LIST) Use L=0 to suppress the listing.
LO=	Listing options: M - storage map O - object code listing (use only if requested by Code 1892) R - cross-reference list S - list source program When more than one is specified, separate them with /, e.g., LO=M/S. (defaults: S / 'M/R/S')
PD=	Print density. 3, 4, 6, or 8 lines/inch may be selected. (defaults: 6/8)
SY	Syntax check only. Do not produce object code. Cuts compilation time roughly in half.

See COBOL5: chapter 12, for additional options.

** Examples **

COBOL5. Defaults to: I=INPUT,L=OUTPUT,LO=S,B=LGO

COBOL5(I=INP,L=OUTP,LO=M/O/R/S,SY,PD=8)

*** Execution Memory and Time ***

*CORE and *TIME parameters can be added to the execution statement. They will display in the dayfile the maximum amount (in octal) of CM and CPU time, respectively, used during execution. For example,

LGO,*TIME,*CORE.

*** COBOL5 versus COBOL 4 ***

COBOL5 and COBOL 4 have substantial differences because they were designed to comply with different versions of the ANSI COBOL Standard. A good condensed list of the major differences is given in chapter 3 in the "COBOL4 to COBOL5 Conversion Aids Reference Manual". Among these differences are:

Duplicate keys are no longer allowed for indexed sequential files.

User-defined labels are no longer allowed.

STRING, UNSTRING, MERGE, and INITIALIZE have been added.

Collating sequence control has been expanded and may be dynamically controlled.

Character code conversion can be specified for a tape in Univac FIELDATA format.

Starting at level 461, files may be shared between subprograms.

Starting at level 461, CALL/CANCEL is activated.

COBOL4

COBOL5

1. Calling a Fortran Subroutine
CALL TEST
2. The abbreviations GR,NGR,LS,NLS
EQ,NQ,GQ,NOT GQ,LQ,NOT LQ are
permitted.
3. No quotes required.

ENTER FORTRAN-X "TEST"

The abbreviations are not permitted. Some forms of equal are not permitted; therefore, two tests may have to be performed.

COBOL5 requires quotes around nonnumeric literal names such as associated with the file name in the ASSIGN clause, the file name in the SPECIAL NAMES SECTION, the information associated with the USE clause of the SELECT statement, etc.

4. Example of SELECT with indexed files:

```
SELECT ABC ASSIGN TO TEST1
      ORGANIZATION IS INDEXED
      ACCESS MODE IS RANDOM
      SYMBOLIC KEY IS TEST-KEY.
```

Indexed files installed as initial.

```
SELECT ABC ASSIGN TO "TEST1"
      USE "ORG = OLD"
      ORGANIZATION IS INDEXED
      ACCESS MODE IS DYNAMIC
      RECORD KEY IS TEST-KEY.
```

Indexed files can be installed as either initial or extended. Extended AAM files are the default. If initial where intended, use the "ORG-OLD" clause.

5. TODAYS-DATE is allowed.

```
In WORKING-STORAGE do:
01 THE-DATE.
   02 THE-DATE-YY PIC XX.
   02 THE-DATE-MM PIC XX.
   02 THE-DATE-DD PIC XX.
01 TODAYS-DATE PIC X(10).
```

```
In the PROCEDURE DIVISION
do:
ACCEPT THE-DATE FROM DATE.
STRING SPACE,
      THE-DATE-MM, "/",
      THE-DATE-DD, "/",
      THE-DATE-YY, SPACE
DELIMITED BY SIZE
INTO TODAYS-DATE.
```

6. INITIALIZE is not a reserved word.

INITIALIZE is a reserved word.

7. Use -FZ on print files

Use "BT=C,RT=Z"

8. No error if PICTURE used with a COMP-2.

An error is produced if a PICTURE is used with a COMP-2.

9. EXAMINE is the valid COBOL statement.

Example:

EXAMINE ident TALLYING UNTIL
FIRST " ".

INSPECT is the valid COBOL statement.

Example:

MOVE 0 TO TALLY.
INSPECT ident TALLYING TALLY
FOR CHARACTERS BEFORE
INITIAL " ".

Note: TALLY is defined in the WORKING-STORAGE. The MOVE 0 TO TALLY is very important.

Example in WORKING-STORAGE:
01 TALLY PIC 9(8) COMP-1.

10. With DISPLAY you cannot use the ADVANCING option.

With DISPLAY you can use the ADVANCING option. This is great for displaying a message on the terminal and then using the ACCEPT command for your response.

11. When reading indexed files, you use MAJOR KEY IS.

When reading indexed files, you use KEY IS.

12. A format of the ADD.....TO statement is allowed without the key word "TO".

The keyword "TO" or "GIVING" is required.

Example:

ADD item1 item2.

Example:

ADD item1 TO item2.

13. REMARKS and NOTE statements permitted.

REMARKS and NOTE replaced by a generalized comment designated by an asterisk in the indicator area.

14. The SKIP verb is permitted.

The SKIP verb has been deleted.

15. COBOL 4 referenced by the control card COBOL. See the CCRM for the correct options.

COBOL 5 referenced by the control card COBOL5. See the CCRM for the correct options.

*** COBOL4 to COBOL5 Conversion Aid ***

A conversion aid program is available to assist in converting programs from COBOL4 to COBOL5. The detailed description of its operation is found in "COBOL Version 4 to COBOL Version 5 Conversion Aid" Reference Manual (CDC 19265021).

Field length required: 77000 CM words.

TAPE11 is the default input source to the translator.

TAPE25 is the default converted output file.

COBOL COPY libraries must be converted via a special routine (See CBLCOP directive in Conversion Aid Reference Manual).

** Convert Source Deck on Cards and Call COBOL5 for Compile **

jobname,CM77000,....	name / code
CHARGE,....	
COPYCR,INPUT,TAPE11.	<-- copy input cards to file TAPE11
REWIND,TAPE11.	
ATTACH,LCS,ID=CSYS.	<-- attach conversion
ATTACH,TAPE10,LCSYNTAX,ID=CSYS.	<-- program files and
LCS.	<-- execute (input on TAPE11)
REWIND,TAPE25.	
COBOL5(I=TAPE25)	
<eor>	
COBOL4 source deck	
<eor>	
UPD,CBLLIST	(lists all COBOL4 source)
CBL	
<eoi>	

** Program to be Converted is on an UPDATE Program Library **
** Translate and Compile **

```
jobname,CM77000,....          name / code
CHARGE,....
ATTACH,OLDPL,....
UPDATE(Q)                      ** select a deck to be translated
ATTACH,LCS,ID=CSys.
ATTACH,TAPE10,LCSYNTAX,ID=CSYS.
LCS,COMPILE.                   ** convert selected program (from
                                ** compile)
REWIND,TAPE25.
COBOL5(I=TAPE25)
<eor>
*COMPILE DECKNAME
<eor>
UPD,CBLLIST
CBL
<eoi>
```

** Convert Source Program from UPDATE PL and Create New PL **

```
jobname,CM77000,....          name / code
CHARGE,....
REQUEST,NEWPL,*pf.
ATTACH,OLDPL,....
UPDATE(Q,S=TAPE11,C=0)         ** select deck (+ *DECK directive)
                                to be converted
UNLOAD,OLDPL.
REWIND,TAPE11.
ATTACH,LCS,ID=CSYS.
ATTACH,TAPE10,LCSYNTAX,ID=CSYS.
LCS.
REWIND,TAPE25.
UPDATE(I=TAPE25,N)             ** see note below
CATALOG,NEWPL,....
<eor>
*COMPILE DECKNAME
<eor>
UPD,CBLLIST
CBL
<eoi>
```

Note: UPDATE will create new PL of translated deck. This will be the only source deck from the oldpl to appear in the newpl.

*** COBOL Compilation Errors ***

When the compiler detects a source language error, it prints out one or more diagnostic messages following the source listing. Each message is associated with an identifying number, a severity code and the line number in which the error occurred. The severity of the diagnostic is indicated by a code letter: T - trivial; U - unconventional; E - error; and C - catastrophic.

A dayfile message giving the total number of diagnostics is printed for each compilation.

*** COBOL Notes ***

1. When passing an array from a COBOL main program to a FORTRAN subprogram, the language-name (FORTRAN-X) should be included in the ENTER statement. If the language-name is omitted, then COMPASS is assumed.

Example: ENTER [FORTRAN-X] routine-name [USING parameter-list]

2. Numeric data format.
COMP-1 items for COBOL 4 and COBOL 5 are not the same.

COMP-1 in COBOL 5 is the same format as FORTRAN Extended integer variables and must not exceed $(2^{*}48)-1$.

COBOL 4 COMP-1 items contain a floating point bias in bits 59 to 48 (unnormalized floating value) and may not be passed to FORTRAN subprograms.

*** COBOL 4 ***

The former COBOL compiler in use at DTNSRDC on the CDC NOS/BE system was COBOL 4. The compiler was designed to be a superset of ANSI COBOL (1968 standard). A control statement option causes the compiler to diagnose as errors any non-ANSI features used. This chapter does not describe the COBOL 4 language (see page i: COBOL4).

*** COBOL 4 Parameters ***

Examples COBOL. defaults to I=INPUT,L=OUTPUT,B=LGO
 COBOL(I=COMPILE,LR,B=0)
 COBOL,I=COMPILE,LRM,U.
 COBOL(E=MAIN,OB=OVER,SUB)

option	action
B=PUNCHB	Gives punched binary decks of all routines; no BCD sequencing at end of cards.
B=lfm	Puts binary images on file <lfm>.
B=0	Specifies no binary output.
B	Defaults to B=LGO.
D	Sets a flag that inhibits loading of the relocatable binary when an E diagnostic is encountered and displays the following message on the dayfile: FATAL COBOL ERROR OR D OPTION IN EFFECT
DB	Causes code to be generated that checks the upper limits of the subscripts to insure that they do not exceed the size of the occurs clause. No subscript checking is done if DB is not selected. This option should be used as a debugging aid.
DB1	Allows the generation of object code which calls the trace feature to trace the flow of the program. This parameter must not be selected unless the trace feature is entered by the COBOL source program. See COBOL, chapter II-14.
E=prog-name	Must be used when the output of a COBOL compilation is to be added to a user library with EDITLIB. The prog-name is the name by which the program will be called when executed from the library. It must be five characters or less and must not duplicate the name in any PROGRAM-ID clause, any ENTRY clause or any implementor name in a SELECT or SPECIAL-NAMES clause. A LOAD, NOGO sequence which creates an absolute file on a file named COBCODE must precede the EDITLIB directives.
I=lfm	Used if COBOL source input is on <lfm> instead of INPUT.
I	Defaults to I=INPUT.

L=lfm Indicates the file on which a normal listing is written. A normal listing includes source statements and major (C and E) diagnostics.

L=0 List output is suppressed except for C and E diagnostics.

L Defaults to L=OUTPUT.

The list parameter key letter 'L' may be suffixed by any combination of the following options to provide features in addition to the normal listing.

X includes T and U diagnostics on list output

R produces a data-name and procedure-name cross reference

C includes source statements copied from a source library in the list output

O produces object code listing (use only when requested by Code 1892)

M produces a data map

N or P Causes non-ANSI features to be diagnosed by the compiler as E type errors.

PD=8 Compile time listings are 8 lines per inch.

PD=6 Compile time listings are 6 lines per inch. (default)

PD Implies PD=8. (print density)

SUB Suppresses certain output from compilation of a subprogram

SUBM Indicates that this COBOL subprogram will be called from a main program written in another language.

U Selects the standard CDC collating sequence (see page A-1) for SORT verb execution and indexed sequential file creation instead of the default ASCII collating sequence.

Z Ignores the COBOL 4 features and produces an object program that is compatible with COBOL 3 files.

For further explanation and other parameters, see COBOL, chapter II-10.

*** Conversion to COBOL from IBM ***

The IBM COBOL referenced is COBOL F, however the concepts apply in most cases to any other version of COBOL.

1. To enclose non-numeric literals, CDC COBOL uses double quotes (") while IBM uses single quote (').

2. All data passed from COBOL main program to COBOL subprograms must be defined in the COMMON-STORAGE section. It is not possible to pass data that is defined in the WORKING-STORAGE SECTION to COBOL subprograms. Non-COBOL subprograms can accept items from working storage via the 'USING' part of the call statement. All data elements in the common must be defined in the same order and be the same length in both main and subprogram.

For example in the main program

```
01 GROUP-DATA-ELEMENT.  
   02 FILLER          PICTURE XXX.  
   02 ELEMENTRY-1     PICTURE X(4).  
   02 ELEMENTRY-2     PICTURE XXX.
```

while in the subprogram

```
01 GROUP-DATA-ELEMENT PICTURE X(10).
```

When compiling a COBOL subprogram to be loaded with a COBOL main program, an additional option 'SUB' must be included on the COBOL control statement to prevent the compiler from writing duplicate data division entries on the LGO file. All files used by a subprogram must be defined in the main COBOL program.

3. In the FILE CONTROL area the SELECT clause is used to relate a lengthy COBOL reference to a file name that fits the conventions established by the computer manufacturer. In CDC systems the 'IMPLEMENTOR NAME' must not exceed 7 characters, may not be bound by quotes, and must be the actual local file name (lfn). CDC conventions are simple and easy to use. There are four special 'implementor names' INPUT, OUTPUT, PUNCH, and PUNCHB. For example -

```
SELECT CARD-INPUT,      ASSIGN INPUT.  
SELECT PRINTER,         ASSIGN OUTPUT.  
SELECT CARD-PUNCH,      ASSIGN PUNCH.  
SELECT CARD-PUNCH-BIN,  ASSIGN PUNCHB.
```

All records read from CARD-INPUT will be the card images from the card reader. All records written to PRINTER will be printed on any available printer after job termination. All records written to CARD-PUNCH or CARD-PUNCH-BIN will be punched after job termination.

3. continued.

In IBM COBOL, all files SELECTed must be on DD control cards even if empty and used only during program check-out runs. With CDC CYBER, external control statements are not always required. If a named input file need not be present when the object program is executed, the word optional must be specified between the file name and the word select. When an optional file is not present, the first read attempt will immediately satisfy the 'AT END' condition and control will pass to the procedure to be executed 'AT END'. If one or more SELECTed files are to be written, they will be created by the system on mass storage. All disk output files, except the special files OUTPUT, PUNCH, and PUNCHB will be released by the system upon job termination unless provision is made to catalog them.

4. As in IBM COBOL an SD (sort description) is required when the SORT verb is to be used. In addition, CDC COBOL requires a select clause for the object of the sd entry.

For example -

SELECT SORT-FILE, ASSIGN TEMP.

SD SORT-FILE, DATA RECORD IS SORT-RECORD.
01 SORT-RECORD.
02 DATA-ELEMENT-1 PICTURE XX.
etc.

5. In the file description entries 'RECORDING MODE' may not be F, U, or V. The options are decimal or binary. The use of binary results in greater efficiency when writing NOS/BE tapes which have a format peculiar to the CDC CYBER. If the 'RECORDING MODE' clause is not used, the compiler defaults to decimal. Tapes written as decimal and read as binary will cause tape read errors (and vice versa). The 'RECORDING MODE' clause has no meaning when the file is a disk or mass storage file.

6. When the 'LABEL RECORDS ARE STANDARD' clause is used, the CDC compiler assumes a tape label that has previously been defined with a NOS/BE system LABEL control statement. The opposite is true in IBM COBOL where all disk files must be described as 'LABEL RECORDS ARE STANDARD'.

7. The 'BLOCK CONTAINS xxx RECORDS/CHARACTERS' clause may not have the same meaning in CDC COBOL as in IBM. Omission of the BLOCK CONTAINS clause will result in a block size best suited to the NOS/BE device to which the file is assigned.

For S and L tapes, the BLOCK CONTAINS xxx RECORDS clause provides the most efficient operation.

8. The first entry in the 'PROCEDURE DIVISION' must be a paragraph name. If not, a diagnostic message will be generated.

For example -

```
PROCEDURE DIVISION.  
BEGIN-PROGRAM.  
    OPEN INPUT, ....
```

9. Printer carriage control is achieved by the special characters that appear in the first position of the record to be printed. CDC has the following conventions:

```
1    page eject  
blank spacing one line  
0    spacing two lines  
-    spacing three lines  
+    no advance
```

Some other carriage control characters are less frequently used.

When using the clause 'WRITE xxxx AFTER ADVANCING x LINES', where <x> is a number greater than zero, the carriage will space the indicated number of lines. Page ejection however requires a '1' in the first position of the output line. The IBM COBOL 'AFTER ADVANCING 0 LINES' must be changed. For example:

```
WRITE xxxx AFTER ADVANCING SKIPO.
```

where SKIPO is defined in the SPECIAL-NAMES paragraph.

```
SOURCE-COMPUTER.  CYBER-176.
```

```
OBJECT-COMPUTER.  CYBER-176.
```

```
SPECIAL-NAMES.
```

```
    '1' IS SKIPO.
```

10. Quotes must be removed from the 'IMPLEMENTOR NAMES' or 'EXTERNAL-FILE-NAMES' in the SELECT clauses. Quotes are not needed on entry point names used in CALL or ENTER statements. Quotes are not needed on the 'PROGRAM-ID', which may be up to 30 characters long, although all but the first 7 will be ignored by the compiler.

11. Input and output procedures are slower than the 'USING' and 'GIVING' options, but the input and output procedures allow some manipulation of data. If input procedure or output procedure is specified, the user must provide such procedures in the form of sections with section headers.

12. All variable subscripts should be defined as 'COMPUTATIONAL-1' independent data elements. In addition, variable subscripts should be incremented by similarly defined elements. For example, to increase variable subscript 'A' by one, use:

```
77 A PICTURE S9, COMP-1, VALUE ??.
77 ONE PICTURE S9, COMP-1, VALUE 1.
ADD ONE TO A.
MOVE DATA-NAME (A) TO ....
```

If variable subscripts are defined as numeric display items, the compiler generates object code to convert the items to computational-1, perform the subscripting operation, and then convert back to display code. When a fixed subscript is used, it may be in numeric display code since the compiler converts it to 'COMPUTATIONAL-1' and it is not altered by program execution.

13. Any IBM program using indexed sequential (ISAM) will require some conversion to use CDC indexed sequential file organization. (see COBOL, chapter II-2)

14. COBOL debugging aids do not include the special verbs TRACE, EXHIBIT, EXHIBIT NAMED, which are IBM extensions. See COBOL, chapter II-14.

15. The LFN parameter on the FILE statement allows file name substitution at execute time. For example, a relocatable COBOL object program was compiled with

```
SELECT TRANSFILE ASSIGN TO INPUT.
```

to substitute a previously cataloged disk file for the file INPUT without recompiling, the following control statements could be used:

```
ATTACH,BIN,COBOLBIN,ID=xxxx.
ATTACH,FIL,DISKTRANSFIL,ID=xxxx.
FILE,INPUT,LFN=FIL.
LDSET,FILES=INPUT.
BIN.
```

Note: The file to be substituted must have compatible Record Manager block and record types or other parameters will be required.

Note: Do not attempt to change the name of an existing SIS or direct access file.

*** COBOL 4 Notes ***

1. All COBOL 4 programs will use Record Manager for input/output functions.

COBOL 4 uses the 'BLOCK CONTAINS' clause to determine the Record Manager block type 'BT' and the 'RECORD CONTAINS' clause to determine the Record Manager record type 'RT' to be assigned.

When processing files on NOS/BE devices with a COBOL 4 program, the 'BLOCK CONTAINS' clause should be omitted or stated as 'BLOCK CONTAINS nnn CHARACTERS' (where nnn is the PRU size of the NOS/BE device being used) for efficient use of the device.

When processing files on S or L tapes, the 'BLOCK CONTAINS' clause should be stated as 'BLOCK CONTAINS nnn RECORDS'.

2. NOS/BE unit record device files must be processed with a zero-byte terminated record type designation, 'RT=Z'. Other files may need to be described as containing zero-byte terminated records. Any file which originated on the card reader but no longer has the logical file name INPUT or any file whose ultimate destination is to be OUTPUT or PUNCH should be designated as containing zero-byte terminated records.

COBOL 4 automatically forces 'RT=Z' for any file assigned to INPUT, OUTPUT, PUNCH, or PUNCHB in the SELECT statement. Should the programmer need to designate Z-type records on other than these files he must choose one of the following means--

- a) A '-FZ' appended to the local file name in the 'SELECT' clause forces Z-type records and overrides the block contains clause.

Example--SELECT MYDISKFILE ASSIGN TO DISK1-FZ.

Note--The local file name used on NOS/BE control statements does not use the '-FZ', i.e., the ATTACH would read--

ATTACH,DISK1,ANYFILE,ID=xxxx.

- b) Use of FILE and LDSET statements for the local file name of the file with Z-type records will override the COBOL generated block and record types.

Example--

```
jobname,.... name / code
CHARGE,....
COBOL(LR,U)
ATTACH(DISK1,ANYFILE,ID=xxxx)
FILE(DISK1,BT=C,RT=Z)
LDSET(FILES=DISK1)
LGO.
<eor>
(COBOL source program)
<eoi>
```

3. In COBOL 4, it is fatal to open a file as "INPUT" if the file does not exist. If 'OPTIONAL' is specified, the 'OPEN' is legal and the 'AT END' is taken on the first read.
4. The value specified in the 'ADVANCING' phrase of the 'WRITE' verb causes the exact number of lines to be advanced on the printer. The combination of a 'WRITE linex AFTER ADVANCING nnn LINES.' followed by a 'WRITE linexx BEFORE ADVANCING nnn LINES.' will result in linexx being written over the top of linex.
Note--Do not make the AFTER ADVANCING changes to your program if you are using the Z parameter on the COBOL statement.
5. Under NOS/BE, the system default collating sequence is the ASCII6 collating sequence (numerics precede alphabets). To override the default collating sequence and insure that a program utilizes the CDC standard COBOL collating sequence include a 'U' parameter on the COBOL control statement.

Example---COBOL(U)
COBOL(I=LFN,LR,U)

6. All EOR's (7/8/9 cards) in file INPUT are considered to be end-of-file markers by COBOL 4.
7. To use a file that has been closed under COBOL 4, another open statement is required.

8. Under NOS/BE, no subscript value checking is done unless DB is selected on the COBOL control statement. When DB is not selected, the program executes faster because less code has been generated. However, no DB can lead to undetected errors if you have relied on the subscript diagnostic message in the past. A good idea might be to run with DB selected during test phase and not selected for the production runs.
9. COBOL 4 initializes WORKING-STORAGE areas which do not have VALUE clauses to blanks. If a program expects an uninitialized area to contain other than spaces the program must explicitly cause that area to contain the correct value.
10. Under NOS/BE, when a field being moved to a numerically edited field contains a non-numeric character the message 'NON-NUMERIC IN FIELD TO BE EDITED' is issued and the run terminated. If for some reason the run must be allowed to continue with the non-numeric error, an 'EE=nn' parameter may be specified in the execution call to allow a maximum number of non-fatal execution errors. When the count specified by the integer is exceeded and an error occurs, the run is terminated.

Example--LGO(EE=10)

11. In COBOL 4, the declarations 'RECORD CONTAINS DEPENDING UPON data-name' and 'OCCURS DEPENDING UPON data-name' are not acceptable if the data-name is COMP-1 or COMP-2. The data-name must be numeric and must be 6 or fewer characters in length.

*** Sample COBOL Deck Setups ***

** Compile, No List, and Execute **

```
xxxxSTS,CM100000,T100.                name / code
CHARGE,....
COBOL(L=0)                             <-- l=0 suppresses source listing
LGO.                                   <-- load and execute from file 'LGO'.
<eor>
  (COBOL source deck)
<eor>
  (data deck as needed)
<eoi>
```

** Compile and Execute Using a Data Tape **

```
xxxxST2,CM100000,T100,MT1.            name / code
CHARGE,....
COBOL(B=OBJECT)                       <-- see COBOL statement options above
LABEL,DATAIN,R,D=HY,L=xxxxDATAIN,VSN=CB7777,NORING.
OBJECT.                               <-- load and execute from file 'OBJECT'
<eor>
  (COBOL source deck)
<eor>
  (data deck as needed)
<eoi>
```

** Compile and Execute With Subprograms **

```
xxxxST3,CM61000,T200.                name / code
CHARGE,....
COBOL.                                <-- compile main program
COBOL(SUB)                            <-- compile COBOL subprogram
COBOL,SUB.
LGO.                                  <-- load and execute entire object program
<eor>
  (COBOL main program source deck)
<eor>
  (COBOL first subprogram)
<eor>
  (COBOL second subprogram)
<eor>
  (data deck as needed)
<eoi>
```

** Compile, Catalog Object Code, Execute **

```
xxxxST1,CM100000,T100.                name / code
CHARGE,....
REQUEST,BINLIB,*PF.                    <-- get space for permanent file
COBOL(B=BINLIB)                        <-- COBOL output to file
CATALOG(BINLIB,DAYACCOUNT,ID=xxxx,XR=RDONLY,PW=RDONLY)
BINLIB.                                <-- load and execute file
<eor>
    (COBOL source deck)
<eor>
    (data, as needed)
<eoi>
```

** Execute Cataloged Object Code **

```
xxxxOB4,CM100000,P2.                  name / code
CHARGE,....
ATTACH,DAYAC,DAYACCOUNT,ID=xxxx.
DAYAC.
<eor>
    (data, as needed)
<eoi>
```

*** Overlay for COBOL 4 ***

The 'SEGMENTATION' feature of COBOL is a means of overlaying procedure division code. While this is easy to accomplish, the major disadvantage is that non-procedure division portions of subroutines cannot be overlaid. Hence the (0,0) level contains the data division of the main and all subroutines. To more efficiently utilize memory, FORTRAN overlay techniques allow overlay of whole subprograms.

Given a main program and two subprograms in COBOL, an effective overlay is created by adding to the main link a FORTRAN subroutine to call the overlays, and adding to each overlay level a dummy main FORTRAN program. COBOL cannot be used for the dummy main program in a level. As long as the FORTRAN segments of the program do no file manipulation or arithmetic operations, no CCOMMON or file equivalence is required.

** Compile With Subroutines and Overlay **

```
xxxxTRY,CM61000,T200.                name / code
CHARGE,....
COPYR(INPUT,LGO,C)                   <-- insert first overlay line
COBOL.                                <-- compile COBOL main program
FTN4.                                 <-- compile FORTRAN subroutine
FTN4.                                 <-- compile dummy main program with overlay
COBOL(SUB)                            <-- compile first subroutine
FTN4.                                 <-- compile dummy main program with overlay
COBOL(SUB)                            <-- compile second subroutine
LGO.                                  <-- load and execute
<eor>
OVERLAY(DON,0,0)
<eor>
```

(COBOL source deck which includes 'ENTER FORTRAN-X FORT')

```
<eor>
SUBROUTINE FORT
C   FORTRAN subprogram must not do any I/O, merely call overlays.
CALL OVERLAY ("DON",1,0)
CALL OVERLAY ("DON",2,0)
RETURN
END
<eor>
OVERLAY(DON,1,0)
PROGRAM FORT1
CALL SUB1
C   FORTRAN main program must do no I/O
END
<eor>
```

COBOL subprogram 1

```
<eor>
OVERLAY(DON,2,0)
PROGRAM FORT2
CALL SUB2
C   FORTRAN main program must do no I/O
END
<eor>
```

COBOL subprogram 2

```
<eor>
```

(data, as needed)

```
<eoi>
```

***** ERRORS, DUMPS AND DEBUGGING *****

In this chapter, we have put those things which may help you when something goes wrong: a program terminates abnormally; you have a tape or disk file and don't know what it contains; etc.

*** Error Messages ***

If there are errors of any type, messages appear in the dayfile (listed at the end of each job). Some specific error messages appear as described below.

Compiler errors are printed with the listing of the source code. For FORTRAN, see page 13-21; FTN5, Appendix B; FTN4, chapter III-2. For COBOL, see page 14-8; COBOL, Appendix G.

Object time errors such as bad data format are noted at the end of the output and "FATAL ERROR nn" appears in the dayfile. (see page 13-21)

Certain system errors related to file manipulations (e.g., input and output) generate Record Manager dayfile messages

RM ERROR nnn ON LFN xxxxxxxx

See RMBAM, Chapter 6.

All batch job fatal errors will cause a short diagnostic dump (DMPX) to be printed at the end of your output (page 15-5; NOSBE, 9-1).

** Loader and NOS/BE Errors **

Diagnostic messages are put in the dayfile by the NOS/BE operating system whenever it detects errors. For example, errors found by the loader are identified by "FATAL LOADER ERROR -" followed by additional explanatory diagnostics. See LOADER, Appendix B.

** Tape Parity Errors **

When tape parity errors are recovered by the system, the message contains "RVD". Unrecovered errors print "ERR" and usually abort job.

**** MSS Errors ****

Most user MSS error messages are self-explanatory, however some messages are indicative of system hardware failures:

TIMEOUT ERROR
MSFS COMMUNICATIONS ERROR

indicates a severed connection between the CYBER and MSS. Under INTERCOM, the command is aborted; in a batch job, the operator may request a retry.

<mfn> PERMANENT ERROR

indicates a file which has been corrupted. Notify User Services immediately so that a backup copy can be reloaded.

** Mode Errors **

Mode errors are detected by the central processor and may result from any type of program.

error mode	cause
0	Attempt to execute an illegal instruction. (Probably data has overwritten part of code - this is probably a subscript problem. It may also indicate executing a stop instruction (OOB).)
1	Address out of range. (Usually a subscript error. If the address is 4xxxxx, 5xxxxx, 6xxxxx, or 7xxxxx, then there is a missing subprogram. Subtract 400000 to get the address to the reference to the routine.)
2	Operand out of range -or- Infinite operand -or- Creation of an infinite (such as 'n' divided by zero) (CYBER 176)
3	Modes 1 and 2 occurred simultaneously.
4	Indefinite operand -or- Creation of an indefinite (such as zero divided by zero) (CYBER 176)
5	Modes 1 and 4 occurred simultaneously.
6	Modes 2 and 4 occurred simultaneously.
7	Modes 1, 2 and 4 occurred simultaneously.

Note that on the CYBER 176, creation of an infinite or indefinite by any computation (including integer division by zero!) causes the program to terminate immediately. On the others, it simply causes an "invalid" number to be generated; the program will not terminate until the "invalid" number is actually used in a (future) computation.

*** Reading Dumps - FORTRAN Debugging ***

The dayfile is the first place to look for errors. System error messages and messages from the operator are listed therein (see FTN5, B-26; FTN4, B-9). Each control statement executed is listed, as is the time.

Execution errors generally are of three types:

FORTRAN errors - detected in FORTRAN object time routines

NOS/BE errors - detected by the system

mode errors - detected by the CPU

(see page 13-20, 15-4)

If fatal FORTRAN compiler errors are found in any subprogram, a line is generated in the dayfile for that subprogram. An attempt to load the resulting invalid binary file (i.e., LGO) will give a loader error, unless "debug" is in use. The FTN compiler generates a reference map for each routine, based upon the value for LO (FTN5) or R (FTN4) in the compile statement. The default, LO=S/A (FTN5) or R=1 (FTN4), gives a map adequate for locating items in a dump (see FTN5, 11-10 ff; FTN4, chapter 12).

The loader generates a directory (memory map) of entry points, the first word address of program units with cross reference, and the location of common blocks, as well as a list of unsatisfied externals (subprograms not found). The total memory required for the loaded program is indicated. For basic (non-segload/non-overlay) loads, the total memory required to load the program is also given on the last line. Locations 0-110B in the user's FL are reserved for the system communication region and loader pointers. Each I/O file has a 35-word file information table and a buffer having a default size of 1001 octal words.

A DMPX is generated whenever a batch program is aborted by the system. This dump includes the contents of all registers, the first 100 words in FL and 100 words before and after the location in the program address (P) register at the time of abort. The fatal error in a program may actually occur several instructions before or after the P address due to the multiple arithmetic units. If a relative dump was planted after an EXIT statement, this will follow the DMPX. Since instructions are 15 or 30 bits in length, each CM word is printed in four columns for easy reading. (see SUG, 8-26)

When a program stops with error mode 2 or 4, attempt to locate the invalid value (page D-2: item 4) in C(Ai) list. Then Ai is the address of the number. Calculate the name of the number using memory map and reference map with octal arithmetic. If C(Ai) is blank, then Ai is the out-of-range address in error mode 1. To back trace where a routine was called, look at the 0400xxxxxx word which is usually the third word after the loader map address of the routine.

*** Job Rerun ***

If machine failure, end of shift, or need for unavailable resources makes rerun of a job necessary, the operator or system deadstart may initiate the rerun.

Any job which has properly cataloged, purged, extended, or renamed a file may not be rerun by operator type-in, but a system deadstart might cause any job to be rerun. Jobs which have MSSTOREd a file may still be rerun, but will then abort if the MSSTORE does not include NA=1.

Whenever possible jobs should be set up so they may be rerun. If it is desired to inhibit the possibility of rerun of a job which may have manipulated tapes to some position where a rerun might destroy the contents, a small file can be cataloged and then purged. This can be accomplished by 'BEGIN,NORERUN.'.

Jobs which have been rerun are charged only for the last run.

*** Abort Processing ***

* CP TIME LIMIT *

When a job aborts on "CP TIME LIMIT", 1 CP seconds (MFF) or 2 CP second (MFE) are allocated for "EXIT" processing. This allows time to catalog files, unload, etc. Only one such time extension is allowed.

* IO TIME LIMIT *

When a job aborts on "IO TIME LIMIT", 100 IO seconds are allocated for "EXIT" processing. No IO limit is enforced unless specified by the user.

* MASS STORAGE LIMIT *

When a job aborts on "MASS STORAGE LIMIT EXCEEDED", enough mass storage is available for taking a DMP, and additionally up to 200 PRUs are available for "EXIT" processing. See User Services if your job really needs more than the maximum mass storage. See also page 5-8: LIMIT.

* MESSAGE LIMIT EXCEEDED *

When a job aborts on "MESSAGE LIMIT EXCEEDED", it has reached the maximum of 1000 messages. Twenty more messages are allowed for "EXIT" processing.

* Reprieve *

When reprieve is used within a job to control abort conditions, the maximum additional allocation is 1 CP seconds (MFF) or 2 CP second (MFE), 100 IO seconds, and 200 PRUs mass storage. Only one such reprieve is allowed.

*** File Dumps ***

The following utilities are available for determining unknown characteristics (blocking, record length, mode, etc.) of files. LISTBIN, PRUDMP, and TAPDMP9 are documented in CLIB/U. The others are described in this manual: EDITLIB (chapter 17), ITEMIZE (page 12-4), TDUMP (document file OTHER). NETED is also helpful if used cautiously.

EDITLIB The 'CONTENT' directive lists information about any or all routines on a binary file. The list includes routine name, date, time, compilation machine, entry points, external references, length of object deck in CM words, type of routine (relocatable/absolute). If the binary file is an EDITLIB library, use the 'LISTLIB' directive.

ITEMIZE NOS/BE utility to list contents of a binary file. Output includes record number, name, length, prefix table for relocatable binary or user library. For sequential UPDATE library, only deck names are listed.

LISTBIN List the contents of a binary file or file of CCL procedures. List includes routine name, length (decimal and octal), type of routine (RELOC/ ABS/OVCAP/.PROC), and, optionally, the comments field of the deck(s). An alphabetical list of the modules may also be printed.

NETED If you don't know what is in a file, print the first five lines. You can usually see if it is a binary, source or data file. CAUTION: if it is a binary file and you print many lines, you may not be able to interrupt it until it has stopped printing.

PRUDMP Octal and/or character dump of disk files.

TAPDMP9 9-track tape dump in either hexadecimal (with optional ASCII and/or EBCDIC translation) or octal (with optional BCD and/or Display Code translation). 7-track tape dump in octal (with optional BCD and/or Display Code translation).

TDUMP Octal and/or character dump of 7-track (odd parity) or 9-track SI tapes (also disk files).

***** UPDATE *****

UPDATE is a maintenance program for creating, correcting, and manipulating libraries of source programs and data.

UPDATE data may be any symbolic information (e.g., source statements for a compiler or data records). UPDATE identifiers usually destroy information in columns 73-80. The UPDATE library is a binary file of card or line images in compressed format. The history information retained in UPDATE files allows you to delete corrections previously made or to restore former (inactive) lines.

A disk program library is normally in random format. The random file (faster access but larger PRU space) stores each deck as a logical record with deck list, directory, and index following the last deck. The sequential format (required for tape files) contains one binary record in which the deck list and directory occur first.

UPDATE compile files are 90-character unit records unless the '8' option or *WIDTH directive was used. Source files and 'UPDATE,8,....' compile files are 80-character unit records suitable for punching, unless *WIDTH was used.

*** UPDATE Control Statement Parameters ***

Frequently used file identifiers:

letter	default	purpose
C	COMPILE	Default except for B option runs.
C=0		Do not create COMPILE file.
P	OLDPL	Old program library file.
N	NEWPL	Must appear if a new PL is desired.
L	A1234	Several special print options exist.
L=0		Suppress UPDATE printing.
I	INPUT	Use only if UPDATE directives are in a file.
I=0		No directives.
O	OUTPUT	Use only if UPDATE output is to a file.
M	MERGE	Use only to combine 2 old PL's to NEWPL.
K	COMPILE	Cause compile file to contain decks in compile directive sequence - for overlays.
S	SOURCE	File for UPDATE input to resequence decks.

Old PL and new PL may not be same file.

File manipulation:

letter	default	purpose
R		Suppress all rewinds. Otherwise, C,N,P,S rewind before and after.

Options for compile file contents (unless C=0):

letter	purpose
F	All decks (from old PL and UPDATES).
Q	Only decks named on *COMPILE directives when Q and N used, subset these to NEWPL. Default when F and Q omitted: decks modified this run or on *COMPILE directives.
8	Force compile file endpunching into 73-80 for punch.
D	Force compile file endpunching into 81-90 for data.

See page 16-4: *WIDTH.

Options for sequential or random NEWPL:

letter	purpose
A	Convert sequential NEWPL to random disk OLDPL. (no directives are read and no compile file created) (use to restore OLDPL from tape)
B	Convert random disk OLDPL to sequential NEWPL. (no directives are read and no compile file created) (use to backup OLDPL to tape)
W	Disk NEWPL is to be sequential.
default	Disk NEWPL is random; tape NEWPL is sequential.

(See UPDATE, 4-2.)

Examples: UPDATE(P,N)
UPDATE,Q,P=LAST.

*** UPDATE Directives ***

UPDATE requires an input logical record of directives (* in column 1). An EOR is required even if no directives are used, except when parameter A or B or I=0 is used in the control statement.

*COMDECK name2	Store set of common statements once *COMDECK must precede *DECKs that *CALL it.
*DECK name1	Insert new deck (subprogram or group).
*CALL name2	Insert comdeck <name2> into current deck here for compile file.
*ADDFILE filen,namen	Add files (including *DECK directives) to existing library after deck <namen>.
*ADDFILE	Add input decks which follow, after last deck of OLDPL.
*IDENT unique	Initiate correction set.
*INSERT namei.seqno	Insert cards after <seqno> in <namei>.
*DELETE namei.seqno,namei.seq2	Delete cards <seqno> to <seq2> inclusive and optionally insert others.
*DELETE namei.seqno,seq2	and optionally insert others.
*DELETE namei.seqno	Delete card <seqno> and optionally insert.
*COMPILE namei	Copy specific deck to compile file.
*READ filin	Insert record from <filin> without rewinding (not valid for C,P,N,I,O,M,S files).
*SEQUENCE dnamei	Resequence active cards in deck <dnamei>.
*COPY dname,namei.seqno,namei.seq2	Copy in cards from deck <dname> at current insert or delete position.
*MOVE dname1,dname2	Move deck <dname1> after deck <dname2>.
*WIDTH linelen,idlen	Change line and/or ID lengths from default.
*RESTORE namei.seqno	Reactivate a deleted card and optionally insert others.
*WEOR	Cause compile file to include record mark.
*CWEOR	Cause compile file to include record mark if anything was written since the last eor.
*YANK namei	Inactivate a correction set with ID <namei>.
*PURGE namei	Permanently delete a correction set name.
*PURDECK namei	Permanently delete all cards in deck <namei>.

An alphanumeric identifier (maximum of 9 characters) goes on the *DECK or *COMDECK directive for program library creation and on the *IDENT directive for correction or expansion runs. All cards in a particular DECK or COMDECK are sequentially numbered by UPDATE (e.g., SC12.42 is the 42nd card with identifier SC12; card SC12.1 is *DECK SC12). Most directives which manipulate whole DECKs or IDENTs may contain a list (comma separated) or an inclusive list (period separated). These include *COMPILE, *PURGE, *PURDECK, *SEQUENCE.

After *ADDFILE directive only complete decks may occur. Any further correction or insertion cards must be preceded by an *IDENT.

Most directives may be abbreviated (at a time cost) by 1 or 2 letters. Printed output shows full directive with ///// preceding it. For successive corrections in same deck, the 'namei.' may be omitted.

Several compile file directives allow conditional processing using *IF, *ENDIF, *DEFINE.

*** Sample UPDATE Setups ***

** Create Source Program Tape Library, List **

```

xxxxMAK,GE1.                                name / code
CHARGE,....
LABEL,NEWPL,L=xxxxALONELIB,W,D=GE,T=7,VSN=CA9999,RING.
UPDATE(N,W)                                <-- create new UPDATE PL, all decks to compile
RETURN,NEWPL.                              <-- return tape drive to system
COPYSF(COMPILE)                            <-- single space listing with sequence numbers
<eor>
*DECK ALONE                                <-- UPDATE directives and source decks
      (FORTRAN main program)
*DECK ATWO
      (subroutine)
*DECK THREE
      (subroutine)
*DECK FOUR
      (function )
<eor>
<eoi>

```

** UPDATE Old Source Library to New and Execute **

```

xxxxDAE,GE2.                                name / code
CHARGE,....
LABEL,OLDPL,L=xxxxALONELIB,R,D=GE,VSN=CA9999,NORING.
LABEL,NEWPL,L=xxxxTESLIB,D=GE,W,T=7,VSN=CA8888,RING.
UPDATE(N,F,W)                              <-- copy PL with corrections and all to compile
RETURN,NEWPL,OLDPL.                        <-- return tapes to system
FTN5(I)                                    <-- compile from updated file implies I=COMPILE
LGO.                                        <-- execute
<eor>
*IDENT SG1030                              <-- correction must be unique (initials,date)
*INSERT ALONE.57                           <-- correct deck ALONE by insert after card 57
      (FORTRAN statements)
*DELETE FOUR.12,13                         <-- correct deck FOUR replacing cards 12-13
      (new cards to replace deletions - optional)
<eor>
      (data cards, if any)
<eoi>

```

```

*      Alternative Statements if OLDPL and      *
*      NEWPL Names are Chosen by xxxx          *

```

```

LABEL, JOE, L=xxxxALONELIB, R, D=GE, VSN=CA9999, NORING.
LABEL, SAM, L=xxxxTESLIB, D=GE, W, T=7, VSN=CA8888, RING.
UPDATE (P=JOE, N=SAM, F)
UPDATE (P=JOE, N=SAM, F

```

** Create Source Program Disk Library, Execute **

```
xxxxB1,CM61000.                                name / code
CHARGE,....
REQUEST,NEWPL,*PF.
UPDATE(N)
CATALOG(NEWPL,MASTPROGPL,ID=xxxx,XR=RDONLY)
COBOL(I=COMPILE)
LGO.
<eor>
*DECK NAME1
      (COBOL source program)
<eor>
      (data cards, if any)
<eoi>
```

** Create Backup Tape Copy of Disk Source Library **

```
xxxxB2,HD1.                                    name / code
CHARGE,....
LABEL,NEWPL,L=xxxxMASTUP,T=30,D=HD,W,VSN=CJ6666,RING.
ATTACH,OLDPL,MASTPROGPL,ID=xxxx.
UPDATE(N,B)                                <-- change random file to sequential
<eoi>
```

** Total Replacement of Existing Deck **

```
xxxxREP,MT1.                                    name / code
CHARGE,....
LABEL,OLDPL,L=xxxxALONELIB,R,D=HY,VSN=CA9999,NORING.
REQUEST,NEWPL,*PF.
UPDATE(N,E)
CATALOG,NEWPL,ALONEPL,ID=xxxx,PW=KEEP,XR=KEEP.
<eor>
*PURDECK THREE
*PURGE THREE
*ADDFILE
*DECK THREE
      (subroutine source cards)
<eoi>
```

** Select Routines Off Source Subroutine Library **
** and Compile with Own Program **

```
xxxxUP,CM100000,T500.                name / code
CHARGE,....
MSACCES,password.
FTN5.                                <-- compile own programs
MSFETCH,LIBR,NSRDCPM,UN=CSYS.
UPDATE(P=LIBR,Q,L=0)
UNLOAD,LIBR.
FTN4(I,L=0,OPT=1)                    <-- omit L=0 if listing of FORTRAN is desired
LGO.                                <-- execute
<eor>
    (own FORTRAN decks)
<eor>
*COMPILE ARPLN1                      <-- select decks from library
*COMPILE AMKUTM
<eor>
    (data cards, if any)
<eoi>
```

** Add Routines to Existing Library Tape - Print **

```
xxxxMO,T100,MT2.                    name / code
CHARGE,....
LABEL,OLDPL,L=xxxxLASTLIB,R,D=HY,VSN=CA8888,NORING.
LABEL,NEWPL,L=xxxxNEXTLIB,W,D=HY,T=7,VSN=CA9999,RING.
UPDATE(N)
RETURN,OLDPL,NEWPL.
COPYSF(COMPILE,OUTPUT)              <-- list additions to library
<eor>
*IDENT SG12017
*ADDFILE                            <-- insert new decks at end
*DECK NEWONE
    (subroutine source cards)
*DECK ANOTHR
    (subroutine source cards)
<eoi>
```

** Punch Selected Deck **

```
jobname,....                        name / code
CHARGE,....
ATTACH,OLDPL,....
UPDATE,8,Q.
ROUTE,COMPILE,DC=PU,TID=C.          <-- punch at Central Site
<eor>
*COMPILE AMKUTM
<eoi>
```

** Create Program Library from Stranger EBCDIC Source File **

```
xxxXSF,PE1.                                name / code
CHARGE,....
VSN,TAPE1-SLOTxx=STRANG.
REQUEST,NEWPL,*PF.
REQUEST,TAPE1,S,PE,EB,NORING.
BEGIN,COPYBLK.                             <-- reblock to SI file - see CLIB/P
RETURN,TAPE1.                             <-- release tape drive
REWIND,TAPE2.
UPDATE(N)                                  <-- create one deck containing whole program
CATALOG,NEWPL,STRANGEPL,ID=xxxx,XR=KEEP.
COPYSF,COMPILE.                             <-- list library with assigned sequencing
<eor>
*DECK PROG
*READ TAPE2
<eoi>
```

** Subdivide Into Decks - Correct **

```
xxxxSD.                                    name / code
CHARGE,....
ATTACH,OLDPL,STRANGEPL,ID=xxxx.
REQUEST,NEWLIB,*PF.
UPDATE(P,N=NEWLIB)
CATALOG,NEWLIB,STRANGEPL,ID=xxxx,XR=KEEP,PW=KEEP.
FTN4,I.
LGO.
<eor>
*IDENT SG07573
*INSERT PROG.1
    PROGRAM ANY (INPUT=128, OUTPUT=128, TAPE8)
Corrected for CDC FTN Jan 1980 S Good
*INSERT PROG.2
*COPY PROG,PROG.5,PROG.6                   <-- move specification statements
*DELETE PROG.5,6
*INSERT PROG.16
*DECK SUB1
*INSERT PROG.25
*DECK SUB2
*INSERT PROG.31
*DECK SUB3
*INSERT PROG.44
*DECK ZERO
*SEQUENCE <-- subl.zero                   <-- sequence each newly created deck
                                         (subl thru zero)
<eor>
    (data cards)
<eor>
<eoi>
```


** Sample Stranger Source File List **

```
C  program from other computer system
    DATA B/ 18*0./
    DATA BL, PI/ " ", 3.14159/
    REAL B(18), C(6)
    INTEGER BL, C
3  READ 2, IT, J, A
2  FORMAT (A5, I5, F10.2)
    IF (IT .EQ. BL) GO TO 9
    CALL SUB1 (IT, J, C, K)
    IF (N .NE. 0) CALL SUB2 (IT, J, A)
    CALL SUB3 (A, B, K)
    IF (K .EQ. 0) GO TO 3
    WRITE (8) C
9  STOP
    END
    SUBROUTINE SUB1 (ALP, J, C, K)
    REAL C(6)
    IF (J .GT. 6) GO TO 2
    C(J) = ALP
    K = 0
    RETURN
2  K = 1
    RETURN
    END
    SUBROUTINE SUB2 (ALP, J, A)
C  error printout
    PRINT 2, ALP, J, A
2  FORMAT ("ODATA ERROR ", A6, I6, F12.2)
    RETURN
    END
    SUBROUTINE SUB3 (A, B, K)
    REAL B(18)
    DO 5 I=1,18
    IF (B(I) .EQ. 0.) GO TO 3
5  CONTINUE
    WRITE (8) B
    CALL ZERO (B)
    K = 1
    RETURN
3  B(I) = A
    K = 0
    RETURN
    END
    SUBROUTINE ZERO (B)
    REAL B(18)
    DO 1 I=1,18
    B(I) = 0.
1  CONTINUE
    RETURN
C  use this deck as input to examples on preceding page
    END
```

***** EDITLIB *****

Through the EDITLIB utility program, you can create a library of binary routines or control statement (CCL) procedures. EDITLIB can handle the binary output of any compiler or core image (absolute) modules (see page 8-1). Generally, there cannot be more than one relocatable main program module in a library; there can be more than one core image module. Absolute files created by SEGLOAD cannot be processed by EDITLIB.

Relocatable modules in a user EDITLIB library are suitable for automatic loading by loader. A library is made available to the loader by one of the following loader control statements:

```
LIBRARY,libname.      where <libname> is the lfn of the library
LDSET(LIB=libname)
```

The lfn of a library may not be the same as any entry point (routine name) in the library.

A procedure in a library may be called (executed) by name, that is,

```
procnam,....
```

instead of

```
BEGIN,procnam,.....
```

(see page 6-1).

EDITLIB is used to add, delete or replace routines, modify selected parameters, and provide statistics about the library contents.
(See NOSBE, 4-35; SUG, chapter 7)

*** EDITLIB command ***

The format of the EDITLIB command is:

```
EDITLIB(I=lfndir,L=lfnlist)
```

Both parameters are optional.

```
lfndir  - file containing directives (default: INPUT)
lfnlist - file to contain listable output (default: OUTPUT)
          (L=0 will suppress the list)
```

*** EDITLIB Directives ***

EDITLIB directives appear in columns 1-72, one directive per line and have one of the following formats:

keyword.
keyword(<parameters>)

Directives fall into six categories: creating a library; modifying a library; manipulating files; changing library format; listing statistics; miscellaneous.

In the descriptions below, <prog> is a routine name or range of routines and may have one of the following forms:

*	All routines from current position to end-of-file. Note that the first use of * on a file (e.g., REPLACE(*,lfn)) will rewind the file. Later uses within the same EDITLIB will not.
prog1	Specific routine 'prog1'.
prog1/prog2/prog3	Specific routines 'prog1', 'prog2', 'prog3' (ADD and REPLACE directives only - entire file is searched).
prog1+prog2	Routines 'prog1' thru 'prog2', inclusive.
prog1-prog2	All routines except 'prog1' thru 'prog2', inclusive.
*+prog2	All routines from current position thru 'prog2', inclusive.
prog1+*	All routines from 'prog1' to end-of-file.

**** Creating a Library ****

LIBRARY(libname,NEW) Must precede all other directives except
file manipulation and comments.

ADD(<prog>,from) Add specified routine(s) of file <from>.

FINISH. Required to terminate library creation.

**** Modifying a Library ****

LIBRARY(libname,OLD) Must precede all other directives except
file manipulation and comments.

ADD(<prog>,from) Add specified routine(s) of file <from>.
If a routine already exists, an error msg
is issued and the routine is not added.

REPLACE(<prog>,from) Existing routine(s) are replaced. If a
routine does not exist, a message is issued
and the routine is added to the library.

DELETE(<prog>) Specified routine(s) are deleted.

SETAL(<prog>,level) The access level of specified routine or
range of routines is changed to <level>.
To be accessible by control statement or
from Intercom, <level> must be odd.
Default: 0. Generally, use AL=0 for
subprograms; AL=3 for programs.

FINISH. Required to terminate library modification.

**** File Manipulation ****

File manipulation directives may appear anywhere in the directive
record. After an EDITLIB, a random library is rewound; a sequential
library is at end-of-file.

REWIND(lfn) Rewind file <lfn>.

REWIND(lfn1/lfn2/.../lfnn) Rewind all files named.

SKIPF(n,lfn) Skip forward <n> decimal records on file
SKIPF(<prog>,lfn) <lfn>. Skip forward to beginning of named
program on sequential file <lfn>.

SKIPF(n,lfn,f) Skip forward n decimal files on multi-file
<lfn>.

SKIPF(n,lfn) Skip back <n> decimal records on file
SKIPB(<prog>,lfn) <lfn>. Skip back to beginning of named
routine on sequential file <lfn>.

SKIPB(n,lfn,f) Skip backward <n> decimal files on
multi-file <lfn>.

**** Changing Library Format ****

A disk resident library is in random format; a tape resident library is sequential. To change formats, use one of:

RANTOSEQ(rlfn,slfn)	Disk resident random library, <rlfn>, is changed to tape resident sequential library, <slfn>.
SEQTORAN(slfn,rlfn)	Tape resident sequential library, <slfn>, is copied to disk library, <rlfn>, which will be random. <slfn> is rewound after each copy.

These directives may not appear between LIBRARY and FINISH.

**** Listing Statistics ****

A list of information about any or all routines on a library file or any binary file of routines is obtained by the LISTLIB or CONTENT directives. The list includes:

- program name
- date, time and compilation machine
- entry points
- external references
- access level (AL)
- length of object deck in CM words
- type of program (relocatable or absolute)

A library file is listed by:

LISTLIB(<prog>,lfn)

Any file of assembled routines is listed by:

CONTENT(<prog>,lfn)

In both cases, <prog> can have any of the forms described on 17-2.

LISTLIB and CONTENT directives may not appear between LIBRARY and FINISH.

**** Miscellaneous Directives ****

ENDRUN. Stops execution of directives. Normally the last directive. Any directives which follow 'ENDRUN.' will be checked for syntax but will not be processed, nor will any errors in them cause the job to abort.

*/ in col 1-2 indicates a comment and may appear anywhere in the directive record

*** EDITLIB Examples ***

** Create Disk Library of Relocatable Routines **

```
jobname,....          name/code
CHARGE,....
FTN5(OPT=1)           <-- compile subroutines to LGO
REQUEST,MYLIB,*PF.
EDITLIB.              <-- create user library
CATALOG,MYLIB,ID=xxxx,XR=xx. <-- pfn=MYLIB
<eor>
    FORTRAN source for subprogram SUB1
    FORTRAN source for subprogram SUB2
    FORTRAN source for subprogram SUB3
<eor>
LIBRARY(MYLIB,NEW)     <-- establish new library file MYLIB
ADD(*,LGO)             <-- add all routines from LGO
FINISH.                <-- terminate library creation
ENDRUN.                <-- end EDITLIB
<eoi>
```

** Modify Library **

```
jobname,....          name/code
CHARGE,....
FTN5(OPT=1)           <-- compile routines to go into library
                        (or other language compiler)
ATTACH,MYLIB,ID=xxxx,PW=xx. <-- must have extend permission
EDITLIB.              <-- user EDITLIB
EXTEND,MYLIB.         <-- user must extend library
<eor>
    new FORTRAN source for subprogram SUB2
    FORTRAN source for subprogram SUB4
<eor>
LIBRARY(MYLIB,OLD)     <-- set to modify old library MYLIB
REPLACE(SUB2,LGO)      <-- delete old copy of SUB2 from MYLIB,
                        add new copy from LGO
ADD(SUB4,LGO)          <-- add new routine SUB4 from LGO
SETAL(MYPROG,3)        <-- set access level so that MYPROG can be
                        called by control statement or from
                        Intercom
FINISH.                <-- terminate library modification
ENDRUN.                <-- end EDITLIB
<eoi>
```

Note: A single directive REPLACE(*,LGO) could have been used instead of REPLACE(SUB2,LGO) and ADD(SUB4,LGO). Since SUB2 exists, it would be replaced. Since SUB4 does not exist, it would be added to the library.

**** Modify 2 Libraries ****

```

jobname,....
CHARGE,....
FTN5(OPT=1)
ATTACH,MYLIB1,ID=xxxx,PW=xx.
ATTACH,MYLIB2,ID=xxxx,PW=xx.
EDITLIB.
EXTEND,MYLIB1.
EXTEND,MYLIB2.
<eor>
    FORTRAN source for subprogram SUB4
    FORTRAN source for subprogram SUB5
    FORTRAN source for subprogram SUB6
<eor>
LIBRARY(MYLIB1,OLD)
REPLACE(*,LGO)
FINISH.
REWIND(LGO)
LIBRARY(MYLIB2,OLD)
REPLACE(*,LGO)
FINISH.
ENDRUN.
<-- end EDITLIB
<eoi>

```

** Condense a Library **

Deleted and replaced routines are not physically removed from a user EDITLIB library. The user should periodically create a new library from the old one to remove the garbage. Make sure the new library is valid (see CLIB/P: COPYLIB).

```

jobname,....                      name/code
CHARGE,....
REQUEST,MYLIB1,*PF.
ATTACH,MYLIB, ID=xxxx.
EDITLIB.
CATALOG,MYLIB1,MYLIB, ID=xxxx,PW=xx.
<eor>
LIBRARY(MYLIB1,NEW)               <-- establish new library file MYLIB1
ADD(*,MYLIB,LIB)                  <-- add all active routines from library
                                MYLIB
FINISH.                           <-- terminate library creation
ENDRUN.                           <-- end EDITLIB
<eoi>

```

Note: If the library contains core image modules, 'SETAL(<prog>,3)' is required after the ADD directive. <prog> is described on page 17-2.

** Create Absolute Module and Library, then Execute **

```

jobname,....                      name/code
CHARGE,....
FTN5(OPT=1)                       <-- compile program
LOAD(LGO)                         <-- load program from LGO
NOGO,LFN.                         <-- complete loading, put absolute on <lfn>
REQUEST,MYLIB2,*PF.
EDITLIB.                           <-- user EDITLIB
CATALOG,MYLIB2, ID=xxxx,XR=xx,PW=xx.
LIBRARY,MYLIB2.                   <-- establish library for loader to search
NEWPROG.                          <-- execute NEWPROG from MYLIB2
<eor>
    PROGRAM NEWPROG (...)
    (rest of FORTRAN source)
<eor>
LIBRARY(MYLIB2,NEW)               <-- establish new library file MYLIB2
ADD(NEWPROG,LFN,AL=3)            <-- add program NEWPROG from file <lfn>
FINISH.                           <-- terminate library creation
ENDRUN.                           <-- end EDITLIB
<eor>
    data, if any, for NEWPROG
<eoi>

```


** Use Subprograms from a Library **

```
jobname,.... name/code
CHARGE,....
FTN5. <-- program references routine(s) on MYLIB
ATTACH,MYLIB,ID=xxxx.
LDSET(LIB=MYLIB) <-- establish user library for next load
LGO.
<eor>
    FORTRAN source for program which calls at least one of
    SUB1, SUB2, SUB3
<eor>
    data, if any
<eoi>
```

** Using a Program from a Library **

```
jobname,.... name/code
CHARGE,....
ATTACH,MYLIB2,ID=xxxx. <-- attach user library
LIBRARY,MYLIB2. <-- make library available to system
NEWPROG. <-- execute program in library
<eor>
    data, if any
<eoi>
```

*** FORTRAN Overlay in a Library ***

In an overlay program, OVERLAY statements specify the local file name on which the absolute overlay module resides. When an overlay is called, this file is searched until the requested overlay is found.

When an overlay program is added to an EDITLIB user library, three things must be taken into consideration:

- 1) Once an overlay program has been put into an EDITLIB user library, the overlay file name must be the library file name. The subroutine OVLNAME, in library NSRDC, allows an overlay program to run, regardless of the local file name given to the file containing the program, whether it is a library or not. OVLNAME obtains the name of the file currently being executed. This file name is then used in the 'CALL OVERLAY' statements. (See example 1 below; CLIB/N: OVLNAME)
- 2) If an overlay program is replaced in an EDITLIB library, the library should also be condensed (to physically remove the original copy of the program). Many programs will not run unless this is done. (See page 17-11; CLIB/P: COPYLIB)
- 3) Even though the overlay facility allows overlays of a single program to reside on more than one file (for faster location and loading), programs in an EDITLIB user library will reside on one file. Therefore, the 'CALL OVERLAY' statements must all refer to the same overlay file name.

Examples are on the next 2 pages.

** Put an Overlay into a Library **

```
jobname,.... name/code
CHARGE,....
FTN5,OPT=1. <-- relocatable modules are on LGO
ATTACH,NSRDC.
LDSET,LIB=NSRDC.
LOAD(LGO)
NOGO. <-- absolute overlays are on MYFILE
REQUEST,MYLIB,*PF. or ATTACH,MYLIB,ID=xxxx.
EDITLIB.
CATALOG,MYLIB,ID=XXXX. or EXTEND,MYLIB.
<eor>
    OVERLAY (MYFILE, 0, 0)
    PROGRAM MYPROG (INPUT=128, OUTPUT=128, ...)
C
C the following common is used to transmit OVLFILE to
C all subprograms or overlays which need it
C
COMMON /OVLFILE/ OVLFILE
CALL OVLNAME (OVLFILE)
...
CALL OVERLAY (OVLFILE, 1, 0)
CALL OVERLAY (OVLFILE, 2, 0)
...
END
OVERLAY (MYFILE, 1, 0)
PROGRAM AA
COMMON /OVLFILE/ OVLFILE
...
CALL OVERLAY (OVLFILE, 1, 1)
...
RETURN
END
OVERLAY (MYFILE, 1, 1)
PROGRAM BB
...
RETURN
END
OVERLAY (MYFILE, 2, 0)
    etc.
<eor>
LIBRARY(MYLIB,OLD)
ADD(*,MYFILE)
*/
*/ make 0,0 overlay control card callable
*/ from batch and Intercom
*/
SETAL(MYPROG,3)
FINISH.
ENDRUN.
<eoi>
```

** Replace an Existing Absolute Overlay Program and Condense **

```
jobname,....                               name/code
CHARGE,....
FTN5,OPT=1.                                <-- relocatable modules are on LGO
ATTACH,NSRDC.
LDSET,LIB=NSRDC.
LOAD(LGO)
NOGO.                                       <-- absolute overlays are on MYFILE
ATTACH,MYLIB,ID=xxxx.
REQUEST,NEWLIB,*PF.
EDITLIB.
CATALOG,NEWLIB,MYLIB,ID=xxxx.
PURGE,MYLIB.                               <-- to get rid of old library (optional)
<eor>
      (FTN source program)
<eor>
*/
*/  replace program in old library
*/
LIBRARY(MYLIB,OLD)
REPLACE(*,MYFILE)
FINISH.
*/
*/  create new, condensed library
*/
LIBRARY(NEWLIB,NEW)
ADD(*,MYLIB,LIB)
SETAL(MYPROG,3)                            <-- repeat for any other main programs
FINISH.
ENDRUN.
<eoi>
```

***** SUBPROGRAM LIBRARIES *****

System object routines needed by compiled programs are available in system libraries and are discussed below. Additional object routines have been added in libraries of subprograms (18-3), libraries of main programs (19-1) and some miscellaneous programs (19-4). Some of these routines are from VIM (the CDC user's group), from other organizations and some have been written at DTNSRDC. How to obtain Machine-readable documentation and how to use non-current versions of system software are discussed at the end of the chapter.

*** NOS/BE System Libraries ***

NOS/BE has several libraries. No ATTACH is used for most of the system files.

The compilers generate, as part of their output object code, the required LDSET for the normal object library. If a COBOL program calls FTN object library routines or an FTN program calls COBOL object library routines, the user must supply the additional LDSET.

library	contents
AAMLIB	Relocatable routines for Advanced Access Method 2
BAMLIB	Relocatable routines for Basic Access Method 1
BASLIB	Relocatable routines for Basic
BIT8LIB	Relocatable routines for 8-bit subroutines
COBOL	COBOL 4.7 object routines
COBOL5	COBOL 5 object routines
DEBUGLIB	Relocatable routines for CYBER Interactive Debug
DMSLIB	Relocatable routines for DMS-170
FORTRAN	FTN 4.8 object routines (FTN4, chapters 8, 9, I/O; DEBUG)
FTN5LIB	Relocatable routines for FORTRAN 5

NUCLEUS	Standard compilers (e.g., COBOL5, FTN5) System utilities (e.g., COPYCF, UPDATE, AUDIT) Intercom commands (e.g., BATCH, LOGIN, XEQ)
PLILIB	Relocatable routines for PL/I
RUN2P3	RUN FORTRAN 2.3 object routines
SRTLIB	Relocatable Sort/Merge routines used by COBOL and FORTRAN
SYMLIB	Relocatable routines for SYMPL and Common Memory Manager
SYSIO	Record Manager routines, internal I/O macros, CPC routines, FORM object routines and SYMPL object library (replaced at level 508 by AAMLIB, BAMLIB, BIT8LIB, DBUGLIB, DMSLIB, PLILIB, and SYMLIB)
SYSLIB	Relocatable system routines formerly in NUCLEUS (starting at level 508)
SYSOVL	Higher level overlays of compilers in NUCLEUS

*** Libraries of Subprograms ***

Library files exist for each major package of subprograms. Each will require ATTACH and LDSET/LIBRARY statements to initiate automatic subroutine loading. Programs using routines from any of these libraries must have been compiled using NOS/BE or SCOPE 3.4.

To access any of these public libraries:

```
* ATTACH,libnam.      where libnam is a library described below
.
.                    prepare user object program
.
LDSET(LIB=libnam)      or LDSET(LIB=libnam1/libnam2) if several
LGO.                  load user program which calls routines
                      on the specified library
```

Scientific libraries currently include: ARLNALG, EISPACK, FUNPACK,
IMSL, LINPACK, MINPACK, MSL,
NSRDC, NSRDC5, SANDIA, SSP

Plotting libraries currently include: CALCFN, CALC3D, CALC936, SCCALC,
TEK48, TEK30

Printer plots are in: IMSL, NSRDC, SSP

Many of the libraries are proprietary and have only the object code available. The others are non-proprietary and may also have the source code on file. An additional copy of each library is maintained on the MSS under UN=CSYS.

** SYSMISC **

Permanent file SYSMISC contains the object routines for FTN 3.0, COBOL 3.0, Sort 3.0, and SIMULA. To use these routines, the job must attach the library prior to loading the relocatable program:

```
...
ATTACH,SYSMISC.
...
LDSET,LIB=SYSMISC/SYSIO.
LGO.                                <-- load old relocatable program
```

* - for the FORTRAN 77 (FTN5) version of IMSL, use:
MSFETCH,IMSL5,UN=CSYS.

**** ARLNALG ****

The Aerospace Research Laboratories (ARL) Linear Algebra Library is a collection of 34 subroutines for solutions to linear systems and determination of eigenvalues and eigenvectors of real symmetric matrices. Some of these routines are specifically optimized for CDC. (See CLIB and the ARL Linear Algebra Library Handbook (TR 74-0106).)

Machine-readable documentation may be listed using:

```
BEGIN,DOCGET,,ARLNALG,,<routine>,OUTPUT,MSACCES=<pw>.  
(output may be routed to print on narrow paper or Xerox)
```

**** CONMIN ****

CONMIN is a FORTRAN program, in subroutine form, for the solution of linear or non-linear constrained optimization problems. The basic optimization algorithm is the "method of feasible directions". The user must provide a main calling program and an external routine to evaluate the objective and constraint functions and to provide gradient information. If analytic gradients of the objective or constraint functions are not available, this information is calculated by finite differences.

While the program is intended primarily for efficient solution of constrained problems, unconstrained functions minimization problems may also be solved. The conjugate direction method of Fletcher and Reeves is used for this purpose.

Contact: P. Matula, Code 1844, telephone (202) 227-1936.

**** EISPACK ****

The ElGenSystem PACKage (version 2) from Argonne National Laboratory is a collection of 70 subroutines to solve eigenvector and eigenvalue problems. Routines in this package are often superior in speed and accuracy to similar routines in other packages. (see CLIB)

Machine-readable documentation may be listed using:

```
BEGIN,DOCGET,,EISPACK,,<routine>,OUTPUT,MSACCES=<pw>.  
(output may be routed to print on narrow paper or Xerox)
```

Reference : Matrix Eigensystem Routines - EISPACK Guide; Smith, Boyle, Garbow, Ikede, Klema, Moler.

** FUNPACK **

A special FUNCTIONal subroutine PACKage from Argonne National Laboratory containing 24 user-callable routines for Bessel functions, Dawson's integral, elliptic integrals of the first and second kind and the exponential integral. (see CLIB)

Machine-readable documentation may be listed using:

BEGIN,DOCGET,,FUNPACK,,<routine>,OUTPUT,MSACCES=<pw>.
(output may be routed to print on narrow paper or Xerox)

** IMSL (proprietary) **

The International Mathematical and Statistical Libraries package (edition 9) contains 517 subroutines in the following areas:

- . Analysis of experimental design data
- . Random numbers, generation and testing
- . Statistics, basic, non-parametric, special functions
- . Regression analysis
- . Differential equations, interpolation, approximation, smoothing
- . Linear algebraic equations
- . Vector matrix arithmetic

Limited machine-readable documentation may be listed using:

BEGIN,DOCGET,,IMSL,,<routine>,OUTPUT,MSACCES=<pw>.
(output may be routed to print on narrow paper or Xerox)

** LINPACK **

LINPACK is a package of 40 subroutines obtained from Argonne Lab. These subroutines analyze and solve classes of systems of simultaneous linear algebraic equations. Beside the single precision package versions for complex or double precision exist. Routines are included for general, banded, symmetric indefinite, symmetric positive definite, triangular, and tridiagonal square matrices plus least square problems and qr and single value decompositions of rectangular matrices. The package also includes 11 basic linear algebra subprograms.

Reference: "LINPACK Users' Guide", J. J. Dongara, J. R. Bunch, C. D. Moler, G. W. Stewart, SIAM, 1979.

Machine-readable documentation may be listed using:

BEGIN,DOCGET,,LINPACK,,<routine>,OUTPUT,MSACCES=<pw>.
(output may be routed to print on narrow paper or Xerox)

** MINPACK **

MINPACK is a package of 11 user-callable FORTRAN subprograms from Argonne Laboratory for the solution of systems of non-linear equations and non-linear least squares problems.

Reference: ANL-80-74

Machine-readable documentation may be listed using:

BEGIN,DOCGET,,MINPACK,,<routine>,OUTPUT,MSACCES=<pw>.
(output may be routed to print on narrow paper or Xerox)

** MSL (proprietary) **

The CDC Math Science Library (Boeing package) contains over 200 numerical mathematical routines covering the following eight areas:

- . Programmed arithmetic
- . Elementary functions
- . Polynomials and special functions
- . Ordinary differential equations
- . Interpolation, approximation and quadrature
- . Linear algebra
- . Probability, statistics and time series
- . Nonlinear equation solvers

MSL, a SCOPE 3.3 product, is available to users but is no longer supported by CDC.

** NSRDC **

NSRDC is a library of DTNSRDC written and/or supported subprograms. Subroutines formerly available only on tape CLIBRARYUPD3 are included. Many subprograms have never been part of any other library. (See also library NSRDC5.)

'NSRDC' has a wide variety of routines including, but not limited to:

- . Roots of polynomials (HELP(complex), PROOT)
- . Special functions (BESSI/J/K/Y, CELLI/ELLI, FRESNEL, GAMMA, Jacobian elliptic)
- . Integration (KUTMER, SIMPUN, DISCOT, Gauss quadrature)
- . Interpolation and curve fitting (fast Fourier transform, harmonic analysis, least-squares and orthogonal polynomials)
- . Eigensystem (VARAH1/2)
- . Simultaneous linear equations (CGAUSS, CMPINV, MATINS)
- . Character manipulation (ADJL/R, ASHIFT/SHIFTA, CENTER, COMPSTR, EXTPRM, GETCHA/PUTCHA, IFINDCH, LASTC, MOVSTR, REPLAC, CHFILL, TRAILBZ)
- . Sorting (ASORT, ASORTMV, SSORT, SSORTI/F/L)
- . Date manipulation (JGDATE, JULIAN, MONTH, NEWDAT, WEKDAY)
- . Time manipulation (ALTIME/ELTIME, IHMS/ISEC)
- . File manipulation (SKPFIL)
- . Debugging aids and dumping (DUMPA, PRTIME, RECOVERD)
- . Plotting (PLOTPR)
- . Extracting information about job (AC, HERE, IDID, JOBNAME, JOBORG, MFRAME, OVLNAME)
- . ASCII I/O

Documentation for many of the routines in NSRDC can be found in CLIB/N. Individual documents from CLIB/N may be printed by:

BEGIN,DOCGET,,NSRDC,,<routine>,OUTPUT,MSACCES=<pw>.

See also page 19-10: footnote.

** NSRDC5 **

NSRDC5 is a library of DTNSRDC written and/or supported subprograms written in FTN5 (FORTRAN 77). These routines may not be used by FTN4 (FORTRAN 66). If both NSRDC5 and NSRDC are used, the LDSET statement should specify NSRDC5 first (LDSET,LIB=NSRDC5/NSRDC).

Routines in library NSRDC5 include:

- . Character manipulation (CHIN, CVCHIN, CVCHOL, CVHOCH, CVINCH)
- . Extracting information about job (AC, HERE, IDID, JOBNAME, JOBOG, MFRAME)

Documentation for many of the routines in NSRDC5 can be found in CLIB/N. Individual documents from CLIB/N may be printed by:

BEGIN,DOCGET,,NSRDC5,,<routine>,OUTPUT,MSACCES=<pw>.

** SANDIA **

This is a collection of routines from SANDIA Laboratories and currently contains:

DE	ordinary differential equation solver (driver)
DEROOT	integrates an initial value problem for ordinary differential equations until a root is located (driver)
STEP	Adam's integration (used by DE and DEROOT but may be called separately)

Reference: See User Services.

**** SHORTIO ****

SHORTIO, a FORTRAN V4 usable subset of I/O routines, can save up to 4000 octal CM. SHORTIO supports the following:

standard FORTRAN formatted READ/WRITE	(BT=C,RT=Z)
FORTTRAN unformatted (binary) READ/WRITE	(BT=I,RT=W)
FORTTRAN BUFFER IN/ BUFFER OUT	(BT=C,RT=S)

This package correctly interfaces with standard system I/O to allow use of FTN4 READMS/WRITEMS. If overlay or segmentation is used, all SHORTIO routines must be in the root segment. No FILE statements may be used with SHORTIO.

To use:

ATTACH,SHORTIO.	
LIBRARY,SHORTIO,FORTTRAN.	<-- reorder libraries
...	
LDSET,USE=\$PUT.Z\$/\$PUT.W\$.	<-- force modules
LGO.	<-- rest of load sequence

**** SSP ****

Version III of the IBM Scientific Subroutine Package contains only single precision routines. All subroutine names are preceded by the letter 'I' (e.g., 'MINV' in the SSP manual is 'IMINV' in the library.) Since many of these routines were optimized for 32-bit words, use with caution on CDC. These routines are no longer maintained by IBM.

Reference: IBM Form Number GH20-0205, System/360 Scientific Subroutine Package, Version III, Programmer's Manual.

*** Plotting Libraries ***

Plotting routines are available for Calcomp pen plotters, Tektronix-compatible terminals, and printer plots.

Several libraries contain the routines for the various Calcomp plotters available.

See Chapter 22 for a discussion of the Tektronix-compatible terminals.

*** Printer Plots ***

See IMSL: USPLH; CLIB/N: PLOTPR; SSP Manual, page 452 (PLOT in the manual and IPLOT in the library).

*** Off-line Plotters ***

Several different off-line plotters are available at DTNSRDC. Software packages for each are discussed below. A device-independent package (DISSPLA), which can use many different plotters, is discussed on page 18-17.

*** Calcomp Plotters ***

The Basic Calcomp Package for on-line Calcomp plotters and off-line plotters on the CDC computers includes the following FORTRAN routines:

PLOT	convert all pen movement specifications from inches to actual plotter commands
PLOTS	initialize PLOT subroutine; set output device number
AXIS	draw an axis line with appropriate scale annotations and title
FACTOR	enlarge or reduce plot size
LINE	plot a series of scaled data points defined by two arrays (x and y)
NUMBER	draw the decimal equivalent of an internal floating-point number
OFFSET	establish transformation factors for subroutine PLOT
SCALE	examine a data array to determine starting and scale values and convert from inches to actual plotter commands
SYMBOL	draw any sequence of alphameric characters
WHERE	retrieve current position coordinates

The calling sequences are as described in "Programming Calcomp Pen Plotters" manual.

For Calcomp 936 and 763, the first call must be

CALL PLOTS (ibuf, nsize, nt)

to define the output unit "TAPEnt". (For Calcomp 936, <ibuf> and <nsize> are ignored and may be 0.) The last call must be

CALL PLOT (x, y, 999)

to close the file.

The Calcomp 936 package is an EDITLIB library and requires:

ATTACH,CALC936.

LDSET,LIB=CALC936. or LIBRARY,CALC936.

The created 7-track tape may also be processed on Calcomp model 1051.

*** Calcomp 936 Pen Plotter ***
(CALC936)

This library contains routines from the Basic Package. In overlay jobs, subroutine PLOT must be in the main link: (0,0) overlay. The calling sequences are as described in "Programming Calcomp Pen Plotters" manual. Additional routines include:

BUFF	internally-called routine which automatically allocates buffer space
DFACT	similar to FACTOR for independent x and y scaling
DWHR/WOFST	similar to WHERE for locating axis coordinates and offsets
NEWPEN	select alternate pen color

These routines may require up to 3000 octal locations.

The output tape is a 7-track, labelled tape at 800 or 556 bpi.

The file name for the plot output file may not be overridden on the LGO statement or the FORTRAN PROGRAM statement. (See page 14-16: item 15 to override)

*** Calcomp Functional Package (proprietary) ***
(CALCFN)

The Calcomp Functional Software Library consists of several categories. These FORTRAN routines generate calls to the basic software requiring the user to also attach and make available the library for the appropriate plotter (CALC936, Zeta).

Business applications

- AXISB - draws an axis with business oriented annotation
- AXISC - draws an axis with calendar month annotation
- BAR - draws bars for bar graph plotting
- LBAXS - draws a logarithmic axis with business annotation
- LGLIN - plots data either in log-log or in semi-log mode
- SCALG - performs scaling for logarithmic plotting
- SHADE - draws shading between designated lines

Drafting applications

- AROHD - draws arrowheads
- ARROW - draws lines terminated with an arrow
- CNTRL - draws center lines
- DIMEN - draws annotated dimension lines
- LABEL - generates annotation between specified points

General applications

- CIRCL - draws circle, arc or spiral
- DASHL - draws dashed lines connecting a series of data points
- DASHP - draws dashed lines to a specified point
- ELIPS - draws an ellipse or elliptical arc
- FIT - draws a curve through three points
- GRID - draws linear grid
- POLY - draws an equilateral polygon
- RECT - draws a rectangle

Scientific applications

- CURVX - plots a function of x over a given range
- CURVY - plots a function of y over a given range
- FLINE - draws a smooth curve through a set of data points
- LGAXS - draws a logarithmic axis with annotation
- POLAR - plots data points, using polar coordinates
- SCALG - performs scaling for logarithmic plotting
- SMOOT - draws a smooth curve through sequential data points

Miscellaneous

- CRVPT - fits a polynomial curve a set of data points using least squares. Plots fitted curve, original data with reference axes and equations of curve.

** Examples **

1. Calcomp 936 plotter

	name / code
xxxx,MT1.	
CHARGE,....	
ATTACH,CALC936.	<-- basic Calcomp 936 package
FTN,OPT=1.	<-- compile user program
VSN,TAPE10=SLOT99=user01.	<-- user's slot tape
LABEL,TAPE10,L=CALC,D=HY,W,RING,IB.	<-- NOS/BE labld tape 800 bpi
LDSET(LIB=CALC936)	<-- select user library
LGO.	<-- load + execute user program
<eor>	
PROGRAM PLOTPGM (...)	
CALL PLOTS (IBUF, 0, 10)	
...	
CALL PLOT (0., 0., 999)	
END	
<eor>	
(data)	
<eoi>	

*** SC4020 to Calcomp ***
(SCCALC)

SCCALC is a plot library which allows a program written with SC4020 calls to plot on the Calcomp 936. Library SC4020 cannot be used for processing at DTNSRDC because we no longer have the SC4020/SC4060 hardware. Instead, three libraries are used: SCCALC, CALCFN and CALC936. See the example below.

Several SC4020 routines not normally referenced by users are dummy routines. No ID frame will be drawn. The option for user-written XMODV and YMODV for non-linear conversion is not available; however, log plots may be drawn.

SCCALC includes entry point names BPLOTV, CHARDC, DLC1 and common block DLCPLT. These names, as well as the basic routine names listed on page 18-11, may not be user routine names.

The output is on magnetic tape 'TAPE48', which may not be overridden on the LGO statement.

** Example **

```
xxxx,MT1,....          name / code
CHARGE,....
FTN,OPT=1.
LABEL,TAPE48,L=xxxxCALC,D=HY,W,RING,VSN=CA9999.
ATTACH,CALCFN.
ATTACH,CALC936.
ATTACH,SCCALC.
LDSET,LIB=SCCALC/CALC936/CALCFN.
LGO.
<eor>
    PROGRAM MYPROG (INPUT=128, OUTPUT=128, TAPE48, ...)
        (rest of FTN source program with SC4020 calls)
<eor>
    (data, if any)
<eoi>
```

*** Calcomp Three-D Software *** (proprietary)
(CALC3D)

The Three-D program automatically draws three dimensional representations of data that can be expressed as a function of two variables. For 3-D subroutines, see DISSPLA (page 18-17).

Surfaces can be drawn transparent, with all lines shown, or as opaque, with hidden lines removed. The surface function can be specified as an external function to be evaluated over a given x and y range, or as z values at the mesh points of a rectangular array.

Three-D can automatically grid irregularly spaced input data. You may insert additional data into the original data array to provide a smoother plot of rough array data. The package provides for automatic orientation of annotation to fit viewing distance and angle. The driver program is controlled by user-specified parameters on input directives. Also stereoscopic pairs can be produced in accordance with the Users Manual.

The Three-D program may be executed by using the catalogued procedure CALC3D. (See CLIB/P)

At DTNSRDC, if field 'INLU' on the 'B' card is specified, it must be '3' (data on TAPE3).

Reference: Three-D - A Perspective Drawing Software System, Calcomp, 1969.

*** DISSPLA (proprietary) ***

The Display Integrated Software System and Plotting Language (DISSPLA) is a package of machine-independent subroutines which may interface with the DTNSRDC Calcomp plotters and Tektronix 401x and 405x terminals or may create a device-independent file for later post-processing on any plotting device.

1. Direct use of DISSPLA in batch jobs requires three EDITLIB libraries:
ATTACH,DISSPLA.
ATTACH,NSRDC.
ATTACH,plotlib. where <plotlib> is CALC936,
TEK30, TEK48, HOUSTON.

When loading the user program, begin the load sequence with:

LDSET,LIB=DISSPLA/NSRDC/plotlib.

Memory requirements will be from 70000 to 120000.

2. For the Calcomp interfaces, the first DISSPLA reference in user program is:
CALL CALCMP (10)
The user must supply his own tape for TAPE10. (see page 18-11)
Only the CALC763 Calcomp interface routine needs 'TAPE10' on the FORTRAN PROGRAM statement.

3. The direct Tektronix interfaces have excessive memory requirements. The first DISSPLA reference in user program is:

CALL TEKTRN (ibaud) where <ibaud> is 30 or
120 or 480.

Under Intercom, the Tektronix is normally used indirectly (see item 4 and page 18-19).

4. When it may be necessary to output a given set of plots on more than one type of plotting device, to allow for interactive viewing on the Tektronix terminals or to allow for later editing of the plot information, a device-independent interface can prepare compressed output on file "PLFILE" for use by any post-processors. Only the first two ATTACH statements of item 1 above are needed. The user provides for "PLFILE" by:

REQUEST,PLFILE,*PF.

To load the user program, begin the load sequence with:

LDSET,LIB=DISSPLA/NSRDC.

"PLFILE" need not be on the FORTRAN PROGRAM statement.

The first DISSPLA reference in the user program is:

CALL COMPRS

Catalog "PLFILE" for later processing with one of the post-processors.

5. After the device interface routine call, the next DISSPLA reference must be CALL BGNPL (n) where <n> is the plot number.
Each plot ends with CALL ENDPL (n)
To properly flush the plot output buffers, the last DISSPLA reference in all user programs must be:
 CALL DONEPL

References:

DISSPLA Pocket Manual
DISSPLA User's Manual

*** DISSPOP: The Post-Processor Option for DISSPLA ***

DISSPOP enables the distribution of DISSPLA-generated plots to different plotting devices. DTNSRDC has post-processors for the following devices: Calcomp-936, Tektronix 401x and 405x (at 300-, 1200- and 4800- baud). Post processors in EDITLIB library "DISPOST" include: POP936, TEK300 (also used for 1200-baud lines), and TEK480.

Calcomp-936	---	POP936
Tektronix (300, 1200)	---	TEK300
Tektronix (4800)	---	TEK480

To access any post-processor a device-independent interface can prepare compressed output on file "PLFILE" to be used by all available plotting devices.

A certain amount of editing of the plot information is often necessary after the application program has generated the plot. Editing features available in "DISPOST" allow:

- . Selective plotting and deletions
- . Windowing
- . Changing and reallocating the position of plots
- . Scaling
- . Superimposing several plots

Commands used to activate editing features are called directives. They are provided to "DISPOST" as card images from the input file, an alternate file specified by the user, or typed in from an on-line terminal; all in free-format. DISSPOP reads directives or expects typed input from a terminal until:

- . A blank card image is encountered
- . An end-of-file is encountered
- . A dollar sign is the last character encountered on the last line

Editing commands are:

DRAW	--	specify plot(s) to be drawn
DELETE	--	exclude plot(s) or range of plots
MODIFY	--	modify size, scale corner, etc.
SIZE	--	scaling to fit an area
SCALE	--	scaling by a factor
CORNER	--	translation (positioning) of plots
OVERPLOT	--	superimposing plots on a microfilm frame or CRT screen
WINDOW	--	drawing selected areas of the plot
UNIT	--	send printed output and error messages to alternate file rather than file OUTPUT
COLOR	--	1, 2 or 3 for black, blue or red pen, respectively.

** Examples **

** Create User PLFILE **

```

jobname,....      <-- memory requirements will be 70K to 120K
CHARGE,....
REQUEST,PLFILE,*PF.  <-- local file name must be 'PLFILE'
FTN.               <-- PLFILE is not required on PROGRAM statement
ATTACH,DISSPLA.     <-- library containing DISSPLA routines
ATTACH,NSRDC.       <-- library used for banner, date, time, etc.
LDSET,LIB=DISSPLA/NSRDC.
LGO.
CATALOG,PLFILE,ID=xxxx. <-- catalog compressed plot file to be used
                        for post-processing with any of the
                        plotlibs described above

<eor>
  PROGRAM MAIN(INPUT,OUTPUT,TAPE10)
  CALL COMPRS        <-- must be first DISSPLA reference in
                        user program
  CALL BGNPL(n)       <-- initialize all plotting routines.
                        n is the plot number.
  .
  .
  CALL ENDPL(n)       <-- ends each plot
  CALL DONEPL        <-- the last call to DISSPLA routine must
                        always be followed by a call to
                        DONEPL to terminate plot generation
                        and to properly flush output buffers
  .
  .
  END

<eor>
  (data)
<eoi>

```

** 'PLFILE' / Calcomp-936 **

```

jobname,MT1.
CHARGE,....
ATTACH,PLFILE,ID=xxxx.
ATTACH,DISPOST.     <-- post-processor library
VSN,TAPE10=CA9999.
LABEL,TAPE10,L=xxxxMYPLOT,D=HY,RING,W,IB.      <-- user-supplied tape
LIBRARY,DISPOST.
POP936.
<eor>
  (directives --- optional)
<eoi>

```


** 'PLFILE' / Tektronix 401x and 405x **
** (300- and 1200-baud asynchronous terminals) **

See chapter 21 for correct operating procedures for Tektronix 401x and 405x terminals. Once logged in:

ATTACH,PLFILE,ID=xxxx
ATTACH,DISPOST
LIBRARY,DISPOST
TEK300

A message will appear on the screen stating that you are in post-processing mode. Enter directives from the keyboard. If no editing is necessary, type a blank, then depress the RETURN key and the first plot in your plot file will be drawn. Continue until all plots are done.

** 'PLFILE' / Tektronix 401x and 405x **
** (4800-baud synchronous terminals) **

See chapter 21 for correct operating procedures for Tektronix 401x and 405x terminals. Once logged in:

ATTACH,PLFILE,ID=xxxx
ATTACH,DISPOST
LIBRARY,DISPOST
SCREEN,80,66
TEK480

A message will appear on the screen stating that you are in post-processing mode. Enter directives from the keyboard. If no editing is necessary, type a blank, then depress the RETURN key and the first plot in your plot file will be drawn. Continue until all plots are done.

***** PROGRAM LIBRARIES AND OTHER ROUTINES *****

*** Libraries of Main Programs ***

Many frequently used programs and major packages of programs have been put into libraries. Each will require ATTACH and LDSET/LIBRARY statements to access the desired program.

To access any of these public libraries:

ATTACH,libname. where libname is a library described below
LDSET(LIB=libname) -or- LIBRARY(libname)
prog. name of program on libname to be executed

The libraries currently available are:

MNSRDC - local scientific programs

UTILITY - a wide variety of local utility programs

** BIMED **

The BIMED package is no longer available. See page 19-5: BIMEDP.

** MNSRDC **

MNSRDC is a library of scientific programs written and/or maintained at DTNSRDC. Documentation for these routines may be found in CLIB/M. To print CLIB/M, use:

BEGIN,MANUAL,,CLIB,M,OUTPUT.

**** UTILITY ****

UTILITY is a library of DTNSRDC written and/or maintained programs. They may be executed by one of the following:

- a) ATTACH,UTILITY.
LDSET,LIB=UTILITY. or LIBRARY,UTILITY.
prog,.... << program to be executed
- b) BEGIN,UTILITY,,prog,.... << for most programs

Documentation for all routines in UTILITY may be found in CLIB/U. Individual documents from CLIB/U may be printed by:

BEGIN,DOCGET,,UTILITY,,<routine>,OUTPUT,MSACCES=<pw>.

*** Miscellaneous Programs ***

The following programs are not in any libraries but must be individually attached (see CLIB):

COPYBFR Recreate a CDC "random file" from data copied earlier to a sequential file, or create a copy of a random file. It may be used to recreate a proper oldpl if COPYBF was used erroneously. To use:

ATTACH,COPYBFR.
COPYBFR,infile,outfile.

COPYBFR does not run on the CYBER 176. See also CLIB/U:
COPYS.

CVT360 A Snobol program to convert double precision IBM FORTRAN programs to single precision CDC FORTRAN V4.

*** Other Programming Packages ***

		Reference
ARRIBA	All-integer programming system for small integer programming problems including several solution algorithms.	H1 UTEX ARRIBA
BOXJENK	Analysis of time-series models using Box-Jenkins philosophy.	1892
CIVCO	Civil engineering computation system: a packaged program designed to calculate complex and compound curves and surfaces in large geometry problems.	1892
ECAP	Electronic Circuit Analysis Program: integrated system of programs to assist in design and analysis of electrical circuits. A graphical version was obtained from NWL.	T4 CODA ECAP
ELBOW	Pipe stress and flexibility calculation.	ORNL
FLANGE	Pipe stress and displacement calculation.	ORNL
LINWOOD	Large-scale multiple linear regression with graphic output.	1892
NOVACOM	A computer program for non-orthogonal analysis of variance and covariance.	NWL TR-2108 NWL TR-2137
STRESS	Structural analysis language. See users and reference manuals for data preparation and program design.	

** BIMEDP (proprietary) **

The BIMEDP-82 BIO-MEDical statistical programs P-series from UCLA accept data with parameter language control similar to SPSS. It replaces all previous versions of BIMED. Most programs will run in CM120000. This is the CDC FTN5 version, which is maintained by Northwestern University.

BIMEDP routines are available on the Mass Storage System and currently include:

BMDP1D	Simple data description
BMDP2D	Frequency count routine
BMDP3D	T test and t-squared routine
BMDP4D	Alphanumeric frequency count routine
BMDP5D	Univariate plotting
BMDP6D	Bivariate plotting
BMDP7D	Description of strata with histograms and analysis of variance
BMDP8D	Missing value correlation
BMDP9D	Multidimensional data description
BMDPAM	Description and estimation of missing data
BMDP1F	Two-way frequency tables - measures of association
BMDP2F	Two-way frequency tables - empty cells and departures from independence
BMDP3F	Multiway frequency tables - log-linear model
BMDP4F	Frequency tables - replaces BMDP1F, BMDP2F, BMDP3F
BMDP1L	Life tables and survival function
BMDP2L	Regression with incomplete survival data
BMDP1M	Cluster analysis on variables
BMDP2M	Cluster analysis on cases
BMDP3M	Block clustering (see BMDQ3M)
BMDP4M	Factor analysis
BMDP6M	Canonical correlation analysis
BMDP7M	Stepwise discriminant analysis
BMDP8M	Boolean factor analysis
BMDP9M	Scoring based on preference pairs
BMDPKM	K-means clustering of cases
BMDQ3M	Block clustering by improved method
BMDP1R	Multiple linear regression
BMDP2R	Stepwise regression
BMDP3R	Nonlinear regression
BMDP4R	Regression on principal components
BMDP5R	Polynomial regression
BMDP6R	Partial correlation and multivariate regression
BMDP9R	All possible subsets regression
BMDPAR	Derivative-free nonlinear regression
BMDPLR	Stepwise logistic regression

BMDP1S	Multipass transformation
BMDP3S	Nonparametric statistics
BMDP1T	Univariate and bivariate spectral analysis
BMDP2T	Box-Jenkins time series analysis
BMDP1V	One-way analysis of variance and covariance
BMDP2V	Analysis of variance and covariance, including repeated measures
BMDP3V	General mixed model analysis of variance
BMDP4V	General univariate and multivariate weighted anova (University of Rochester)
BMDP8V	General mixed model analysis of variance equal cell sizes

To use: MSACCES,password.
MSFETCH,BMDPxx,UN=CSYS.
BMDPxx,....

Reference: "BMDP Statistical Software 1983 (or 1981)", W. J. Dixon,
editor, University of California Press, Berkeley.

BEGIN,DOCGET,,BMDP,,BMDP82,OUTPUT,MSACCES=<pw>.
(20 pages required reading)

*** Other Sources of Programs ***

1. VIM Software Directory (VIM is the CDC Users Group).
2. The Argonne Code Center (USAEC-related) programs.
3. Special purpose complete programs and programming languages which were purchased from other organizations are discussed in chapter 20.
4. A collection of BASIC routines was obtained from NOL.
5. Some medium-speed remote terminals use a Houston plotter package.

Users may contribute routines to our library of programs and subprograms.

*** Program Documentation ***

Documentation for programs, subprograms and procedures can be obtained from Code 1892.1. Some documentation is available from the system and is obtained by using procedure DOCGET (page 6-4), where <lib> might be:

- UTILITY - programs in library UTILITY
- MNSRDC - programs in library MNSRDC
- NSRDC - some routines in library NSRDC
- NSRDC5 - routines in library NSRDC5
- OTHER - some miscellaneous routines
- PROCFIL - public-access procedures

An index to the various libraries is printed in CLIB. Individual machine-readable documents describing many programs, subprograms and procedures are printed in CLIB/N, CLIB/P and CLIB/U.

Procedure DOCGET is also used to print machine-readable documentation for other libraries (EISPACK, IMSL, SPSS, etc.). These documents normally reside on the Mass Storage System and require your MSS access password to use them. See the individual library descriptions earlier in this chapter.

Procedure DOCGET uses UTILITY program GETDOC.

*** Source Code ***

The source code for many routines is available in UPDATE libraries for user modification. The libraries reside on the Mass Storage System. See example on page 16-6 (top).

library	source code location	
NSRDC	MSFETCH,OLDPL,NSRDCPL,UN=CSYS.	<-- non-math
	MSFETCH,OLDPL,NSRDCPM,UN=CSYS.	<-- math
NSRDC5	MSFETCH,OLDPL,NSRDC5P,UN=CSYS.	
UTILITY	MSFETCH,OLDPL,UTILPL,UN=CSYS.	

*** Other Versions of System Software ***

The NOS/BE operating system is currently at level 552; all of the supported software (COBOL, COBOL5, COMPASS, Common Memory Manager, FORM, FTN4, FTN5, SORTMRG, and object routine libraries for COBOL and FORTRAN) is at level 552. Some obsolete and some experimental new versions of supported software are available. Some reside in the permanent file system and have a permanent file name of the form: lvlPRODUCTS,ID=CSYS, where <lvl> is the level number (552, 508, 461). The others reside on the Mass Storage System and have an MSS file name of the form: PRODllvl, where <lvl> is the level number (439, 434, 420, 414, 410, 406, 401, 380). In some cases, 'PRODUCTS' or 'PROD' in the file name is replaced by a reference to the specific product. For assistance, call User Services.

To access one of these, use:

```
ATTACH,PROD,lvlPRODUCTS,ID=CSYS.      -or-
MSFETCH,PROD,PRODllvl,UN=CSYS.        <-- after an MSACCES
LIBRARY,PROD.      -or-      LDSET,LIB=PROD.
```

The <lfm> for a library may not be the same as any entry point (routine name) in the library.

Certain cautions should be observed when using the non-current software:

- . Certain system control statement (such as LABEL and SKIP) cannot be used while the library is active (after the 'LIBRARY, PROD.') because there are subroutines with the same names which will abort the job.
- . Subroutines in user libraries should not have certain system names (such as SPACE, INSERT and many more) because they might not load properly. This is especially true for SEGLOAD.
- . When EDITLIB user libraries are referenced, the load map should be checked to verify that the proper routines were loaded. It is advisable to use 'LDSET,LIB=PROD/<user libs>.'
- . Routines compiled at level 552, 508, 461, 439, 420 or later cannot be loaded by earlier levels. This is because new loader tables were introduced at these levels.
- . Under Intercom, the non-current software should be used by means of a procedure, because 'LDSET,LIB=PROD' must be used instead of 'LIBRARY,PROD'.

** Using Level 552 **

(don't use while 552 products are part of the system)

```
ATTACH,PROD,552PRODUCTS,ID=CSYS.
(LABEL,lfm,...      if needed, must go before LIBRARY,PROD.)
LIBRARY,PROD.
LOADLDR.
FTN4.
LOADLDR.
LGO.
```

** Using Level 508 **

ATTACH,PROD,508PRODUCTS,ID=CSYS.
(LABEL,lfn,.... <-- if needed, must go before LIBRARY,PROD.)
LIBRARY,PROD.
LOADLDR.
FTN. or COBOL. (COBOL 4 is in COB4LIB,ID=CSYS)
LOADLDR.
LGO.

** Using Level 461 **

ATTACH,PROD,461PRODUCTS,ID=CSYS.
(LABEL,lfn,.... <-- if needed, must go before LIBRARY,PROD.)
LIBRARY,PROD.
LOADLDR.
FTN. or COBOL.
LOADLDR.
LGO. <-- see footnote 1

** Using Level 439 **

(MSACCES,.... <-- if not already supplied)
MSFETCH,PROD,PROD439,UN=CSYS.
(LABEL,lfn,.... <-- if needed, must go before LIBRARY,PROD.)
LIBRARY,PROD.
LOADLDR.
FTN. or COBOL.
LOADLDR.
LGO. <-- see footnote 1

** Using Levels 434 and 420 **

(MSACCES,.... <-- if not already supplied)
MSFETCH,PROD,PROD1vl,UN=CSYS. vl=434 or 420
(LABEL,lfn,.... <-- if needed, must go before LIBRARY,PROD.)
LIBRARY,PROD.
LDTEST.
FTN. or COBOL.
LDTEST.
LGO. <-- see footnote 1

Requests for refund or credit due to software failure when using levels 420 will not be honored.

-
1. Routines in library 'NSRDC' should work with the current level of NOS/BE. Some routines have a version for level 461 (and earlier) in library '461NSRDC,ID=CSYS'. If routines are needed from both libraries, use
ATTACH,NSRDC.
ATTACH,N461,461NSRDC,ID=CSYS.
LDSET,LIB=N461/NSRDC.
before the 'LGO.'.

** Using Levels 414 thru 380 **

(MSACCES,.... <-- if not already supplied)
MSFETCH,PROD,PRODlvl,UN=CSYS. lvl=414, 410, 406, 401, or 380
(LABEL,lfn,.... <-- if needed, must go before LIBRARY,PROD.)
LIBRARY,PROD.
...
LGO. <-- see footnote 1 (page 19-10)

Requests for refund or credit due to software failure when using levels
414, 410, 406, or 380, will not be honored.

***** SPECIAL PURPOSE PROGRAMMING LANGUAGES *****

This chapter discusses various programming languages and packages available on the CDC computers at DTNSRDC.

*** ABAQUS ***

ABAQUS is a family of modeling capabilities based on the finite element method, designed to provide solutions to a wide range of mostly non-linear structural problems, and programmed around a common data management structure. The code is designed for production use; that is, it is intended to provide reliable solutions to difficult problems when used by engineers who are responsible for design analysis, and who are not necessarily experts in numerical methods.

The idea of "user friendliness" is a fundamental aspect of the design. This includes minimizing the need for the user to make decisions. For example, the program uses automatic time or load stepping for all its procedures, including static and dynamic stress analysis and heat transfer problems. In addition, the input data concepts are carefully designed for ease of use, the user manuals are clear and unambiguous and the code provides all ancillary support such as mesh generation and comprehensive model and results plotting, as well as extensive data checking.

The program has a comprehensive library of elements (2-D and 3-D solids, shells and beams), materials (isotropic and anisotropic elastic, plastic, creep, etc.) and procedures (static stress, dynamic stress, impact, small or large displacements, steady state and transient heat transfer, eigenvalue extraction, etc.) Enhancements and new capabilities are being developed continuously. The current version is 4-4-53.

The ABAQUS system has two parts: ABAQUSPR, which checks the input data, plots the undeformed structure, if requested, and generates a FORTRAN main program. The second part of the ABAQUS analysis compiles this program and accesses library 'ABAQUS'.

ABAQUSPR requires 162000 CM words. The generated program will require 200000-300000 CM words for a medium-sized problem.

A document describing the required job control statements may be printed by:

```
BEGIN,DOCGET,,OTHER,,ABAQUS,OUTPUT,MSACCES=<pw>.
```

Contact: P. Matula, Code 1844, telephone (202) 227-1936.

*** ALGOL5 ***

Algol5 is no longer supported on DTNSRDC computers.

*** APL (A Programming Language) ***

The APL language had its origins in the book "A Programming Language" by Kenneth E. Iverson (John Wiley and Sons, New York, 1962).

"Because a single line in APL typically expresses what would require many lines in other languages, programs can be written in APL in a fraction of the time with less chance of error."

CDC's APL Version 2.2 is virtually identical to APLUM (APL - University of Massachusetts).

To enter the APL environment, type

APL,ID=xxxx <-- files saved within APL will be CATALOGed
under this ID

For further documentation, consult APL VERSION 2 REFERENCE MANUAL
(CDC publication number 60454000).

*** Automatically Programmed Tools (APT) ***

The APT system is used to prepare punched paper tapes for numerically controlled machine tools. From input language describing a part to be machined, apt generates the detailed commands necessary to direct the motions of a NC machine tool. Using this system, even complex parts can be machined quickly, accurately, and inexpensively.

In APT3, an extension to APT called FMILL is available for machining of arbitrarily shaped three-dimensional surfaces such as found on model propellers and model ship hulls. It accepts as input any rectilinear mesh of x-y-z coordinates. Post processors implemented with APT3 include: GECENT, BENDIX, COLTMC.

A newer version of APT (APT4) is also available on permanent file. There are no post processors installed in APT4. Post processing is possible using an option under APT3.

** APT3 Sample Setup **

```
jobname,CM65200.                name / code
CHARGE,....
COPYCR,INPUT,INFILE.
REWIND,INFILE.
ATTACH,APT3.                    ** APT3 system
APT3,INFILE.                    ** execute APT3
ATTACH,PUNTAP.                  ** paper tape punch program
PUNTAP.                         ** punch paper tape
<eor>
  (APT source cards)
<eor>
  (PUNTAP data card)
<eoi>
```

Reference: APT Reference Manual version 2 (APT III), CDC pub. no. 60174500C.

** APT4 Sample Setup **

```
jobname,CM122000.              name / code
CHARGE,....
ATTACH,APT4.                    ** APT4 system
RFL,122000.                     ** execute APT4
APT4.                           (PUNTAP may be executed here as above)
<eor>
  (APT source cards)
<eoi>
```

Note: If APT4 source cards are in a named file, it will be rewound.

Reference: Automatically Programmed Tooling System (APT IV) version 2 Reference Manual, CDC pub. no. 17326900.

*** BASIC 3 ***

BASIC 3 is an extension of the original Dartmouth Basic, an all-purpose programming language. BASIC 3.5 conforms to ANSI Minimal Basic. Some features of CDC BASIC 3.4 (level 461 and before) were in conflict with ANSI Minimal Basic (X3.60-1978) and were modified in BASIC 3.5.

1. Array lower bounds are 0 (formerly 1). This changed the operation of MAT READ since
DIM A(2,2)
MAT READ A
now reads 9 numbers rather than 4.
2. The default collating sequence is standard (ASCII) rather than native (Display Code).
3. INPUT and MAT INPUT allow reply continuation lines. MAT INPUT is entered as one response rather than a row at a time.
4. Unquoted strings in DATA may now begin with any character except comma or blank.
5. TAB function processing is altered; TAB(x) causes printing in column x.
6. Integer numbers requiring more than 6 digits will print in E-notation.
7. Functions without parameters are allowed.
8. All values being used for subscripts are rounded to integers rather than truncated.
9. Matrix operations may be in-place. Inversion of a singular matrix will return DET=0.
10. Final value of loop control in FOR-NEXT will be first value not used rather than last value used.
11. CID may be used to debug a Basic program.
12. Since a line may be up to 150 characters, no end punching (such as that produced by UPDATE or EDITOR) may be on source lines.

To use BASIC in a batch job:

BASIC,I=1fn1.	<-- executes without creating an object file
BASIC,I=1fn1,B=1fn2.	<-- creates object file on 1fn2 (the default is <u>not</u> LGO)
1fn2.	<-- executes the object program with load map

*** COMPASS ***

The COMPASS assembler is used to maintain many of the NOS/BE system routines for CPU and PPU. Source statements include pseudo-instructions, macros and micros as well as symbolic machine instructions.

*** Checkpoint/Restart ***

Checkpoint dumps may be taken by the programmer periodically during long running jobs (for example, over 30 minutes wall clock time). Then RESTART is used to reinitiate the job from a dump in the event of program or system failure. Checkpoint dumps may be called by a control card or from a user program. (See NOSBE, 4-12, 4-72)

control statement	CKP.	
FORTTRAN Extended	CALL CHEKPTX (ivar)	where ivar=0

Unless the user REQUESTs or LABELs a tape with checkpoint disposition, the system will request a scratch tape for the dumps. The job card must allow for this tape. Each user should provide for his own checkpoint tape. The CK parameter is required on any LABEL or REQUEST statement for a checkpoint reel.

LABEL,CCCCCCC,L=xxxxCKPT,X=CK,D=GE,RING,VSN=CB9999,W.
(use above LABEL statement for initial checkpoint run.
For restart, change parameter W to R.)

The system uses file names CCCCCC, CCCCCCI, CCCCCCM, and CCCCCCO.

If any device set is used in the job, the user must MOUNT the device set before the RESTART statement. If any tapes are used in the job, the user should supply VSN statement(s) (not LABEL or REQUEST) before RESTART.

RESTART,FILEnm,num.
where filenm is checkpoint tape
 (default: CCCCCC)
 num checkpoint dump to restart from
 (default: 1)

If the restart tape is a multi-reel file, it is possible to restart from other than the first tape.

Checkpoints may not be taken in a program using random access (READMS) files unless they are previously cataloged.

If a multi-step job takes a checkpoint between steps of the job, the restart will commence executing control statements at the immediately next step after the 'CKP.' statement. No alteration of INPUT control statements or data records is accepted, as no statements in the restart job stream will be looked at after the 'RESTART.' statement. For a multi-step job to checkpoint only in case of error (after EXIT), a CCL procedure must follow the 'CKP.'.

Checkpoint Examples - FTN

```

PROGRAM SGLONG (INPUT, OUTPUT, TAPE6=OUTPUT, TAPE5=INPUT)
DATA IVAR / 0/
B = TIME (DUM)
DECODE (10, 5, B) IH, IM, IS
ITIME = IH*3600 + IM*60 + IS
C  enter long program loop
    DO 500 K=1,1000
        . . .
        . . .
        . . .
C  take checkpoint dump every 20 clock minutes
5  FORMAT (3 (1X, I2))
    B1 = TIME (DUM)
    DECODE (10, 5, B1) IH, IM, IS
    INOW = IH*3600 + IM*60 + IS
    IF (INOW-ITIME .LT. 1200) GO TO 15
    CALL CHEKPTX (IVAR)
C  recalculate starting time in case of a restart
    B = TIME (DUM)
    DECODE (10, 5, B) IH, IM, IS
    ITIME = IH*3600 + IM*60 + IS
15 CONTINUE
500 CONTINUE
    STOP
    END

```

```

PROGRAM SGLONG (INPUT, OUTPUT, TAPE6=OUTPUT, TAPE5=INPUT)
DATA IVAR / 0/
B = SECOND (DUM)
C  enter long program loop
    DO 500 K=1,1000
        . . .
        . . .
        . . .
C  take checkpoint dump every 360 CPU seconds
    B1 = SECOND (DUM)
    IF (B1-B .LT. 360.) GO TO 15
    CALL CHEKPTX (IVAR)
C  set next time interval
    B = B1
15 CONTINUE
500 CONTINUE
    STOP
    END

```

Note-- By deleting *** lines above, would checkpoint each loop.

AD-A150 162

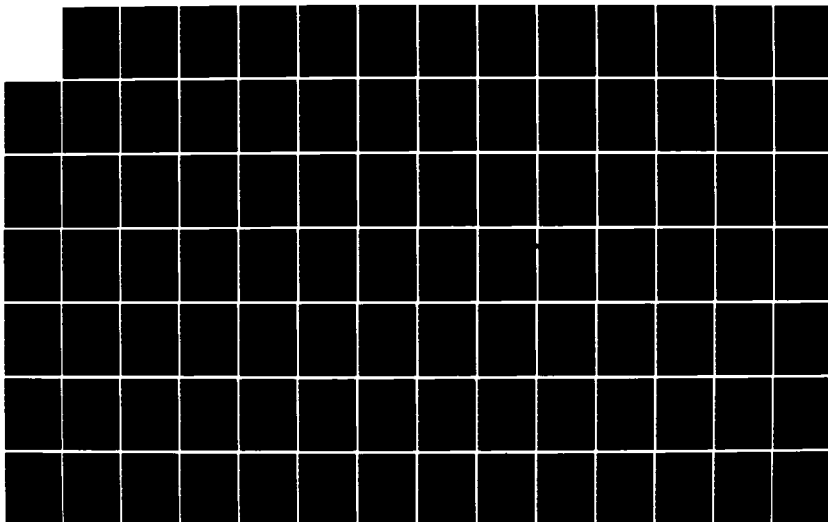
COMPUTER CENTER CDC REFERENCE MANUAL(U) DAVID W TAYLOR
NAVAL SHIP RESEARCH AND DEVELOPMENT CENTER BET..
D V SOMMER ET AL. SEP 84 DTNSRDC/CMRD-84-10

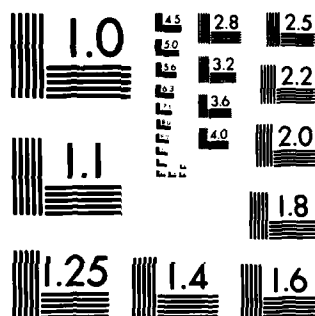
4/3

UNCLASSIFIED

F/G 9/2

NL





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

*** Operator Checkpoint ***

At end of shift or when a long running job must be terminated, the current status of the job may be preserved on tape for restart at a later time by operator action.

1. n.LOCKIN. To LOCKIN the job.
2. n.CHECKPT. To save the current status of job on tape.
3. n.ASSIGN xx Lend the user unlabelled scratch tapes.
Paste on a label with jobname and date.
Hold 24 hours. (e.g. CALMKDR, 6/30/77)
4. n.COMMENT. RESTART JOB USING TAPE . SEE USER SERVICES
n.COMMENT. DROPPING JOB DUE TO -----.
5. n.DROP. To vacate control point.
6. User may restart the job by putting in a short job.

```

xxxx,CMnnnnnn,GEN,Tnnn.      name / code
CHARGE,....
VSN,CCCCCCC=DISPxx.      << see footnote 1
LIMIT,<limit>.          << if needed
      (put in MOUNT/VSN cards for packs/tapes used by job)
RESTART,CCCCCCC,1.
<eoi>

```

where Tnnn is time required to complete job
 CMnnnnnn is core required to run the job
 GEN must allow for mounting the restart tape
 (may be another density)
 DISPxx is as pasted on the scratch tapes.

Final output of the restarted job will include all output computed and printed before the restart was begun.

 1) If checkpoint required more than one tape, use:

VSN=CCCCCCC=DISPxx/DISPxy/....

*** Data Management System 170 (DMS170) ***

DMS170 is Control Data's implementation of a comprehensive data management system. Among the major components of DMS170 are:

Data Description Language (DDL)

This compiler describes the data by creating schemas (which describe the organization and full record content of the data base) and sub-schemas (which describe sets of data elements to be operated on via COBOL or Q/U). It is CDC's implementation of the Codasyl Data Base Task Group Data Description Language.

Query Update (Q/U)

This special English-like language allows retrieval, manipulation and updating of data base elements and includes report writer capability. It can access data using DDL sub-schema or directly using Q/U description of file. It allows batch or interactive sessions.

CYBER Database Control System (CDCS)

This handles data mapping, data validation, transaction logging, and data base procedure linking.

Data Base Utilities

Utilities are provided for logging and data base recovery.

DMS170 provides high level Data Base Management features such as data security, transparency of data organization to end user, logging and recovery, etc., while using standard CDC Record Manager techniques for actual data file organization.

References for further information.

DMS170 Data Administrators User's Guide	60499100
DDL 3 Reference Manual vol. 1	60481900
DDL 3 Reference Manual vol. 2	60482000
DDL 3 Reference Manual vol. 3	60482100
Query Update 3 Reference Manual	60498300
Query Update 3 User's Guide	60387700
CDCs 2 Reference Manual	60481800
Data Base Utilities	60498800

*** GPSS (proprietary) ***

General Purpose Simulation System (GPSS) V, is a generalized simulation package. It is compatible with IBM System/360 GPSS V.

To use version 1.3 (needs CM70000):

```
MSACCES,<password>.
MSFETCH,GPSS,UN=CSYS.
GPSS,<parameters>.      <-- use FX parameter for fixed format
```

To use version 1.2 (needs CM130000):

```
MSACCES,<password>.
MSFETCH,GPSS,GPSS12,UN=CSYS.
GPSS,<parameters>.      <-- use FX parameter for fixed format
```

GPSS is not supported by CDC or the Computer Center. Requests for credit because of software failure will not be honored.

*** MIMIC ***

MIMIC is a digital/analog simulation language for solving systems of ordinary differential equations. MIMIC, a continuous system simulator, is a parallel language since the statements defining a program may be written in any order. The original language was developed by the Air Force. To use:

```
MSACCES,<password>.
MSFETCH,MIMIC,UN=CSYS.
MIMIC.
```

MIMIC is not supported by CDC or the Computer Center. Requests for credit because of software failure will not be honored.

Reference: Mimic Simulation Language, CDC publication number 44610400D.

*** NASTRAN ***

NASTRAN is a general purpose finite element structural analysis program, capable of performing a wide range of analysis on models of complex structures. Some of the capabilities of NASTRAN include static stress analysis, natural frequency analysis, buckling analysis, frequency response analysis and transient response analysis.

The version of NASTRAN currently stored on system permanent files is April 1984.

Execution of the program may be invoked by the following sequence of control cards:

```

jobname,....    with CM, MT, PE, T  set as needed for size of data
CHARGE,....
LIMIT,7777.
VSN,....        as needed for tapes
* LABEL,NPTP,D=PE,RING,....    optional-for runs with CHPNT yes
* LABEL,OTPT,D=PE,NORING,R,.... optional-for restarts
REQUEST,PLT2,*PF.             optional-for plotting (Calcomp 936)
MAP,OFF.
ATTACH,NASTRAN.
BEGIN,NASTRAN,NASTRAN,nnnnnn.  nnnnnn is job card CM
* EXIT,U.
CATALOG,PLT2,....             next 6 cards for plotting option
REWIND,PLT2.
ATTACH,PLTTRN,PLTTRN936,ID=CAMY,MR=1.
UNLOAD,TAPE3.
LABEL,TAPE3,L=...,D=HY,W,RING,IB,VSN=....
PLTTRN.                  writes Calcomp tape
<eor>
      (data cards)
<eoi>

```

See "NASTRAN Theory the Application Course Supplement" (DTNSRDC/CMLD-81-05).

To allow PLT2 to be written to disk rather than tape, it should be in the files parameter of the NASTRAN data card. For example:

NASTRAN CONFIG=6,FILES=(NPTP,OTPT,PLT2)

Undeformed structural plots can also be made interactively using the "staging" system.

See page 22-10: NASTEK, an interactive graphic postprocessor.

NASTRAN requires a minimum of 160000 octal of central memory. However, memory requests should be adjusted to problem size for efficient solution.

* - NPTP AND OTPT may be permanent files instead of tapes. Replace LABEL with REQUEST,...,*PF and add CATALOGs after EXIT,U.

The "BANDIT" preprocessor, for automatic grid point resequencing to reduce the matrix profile, is included in NASTRAN. It will be executed automatically unless bypassed by including the option 'BANDIT=-1' on the NASTRAN data card.

BANDIT may still be executed before NASTRAN by replacing the "BEGIN,NASTRAN" control statement by the following sequence of control statements:

```
ATTACH,BANDIT,ID=CAEE.  
RFL,nnnnnn.  
BANDIT.  
BEGIN,NASTRAN,NASTRAN,nnnnnn,TAPE8.
```

See "BANDIT Users Guide" (TM-184-77-03) for more information.

Additional information on the NASTRAN program and related programs is available from the Numerical Structural Mechanics Branch, Code 1844, telephone (301) 227-1938 or (301) 227-1660.

*** OMNITAB ***

OMNITAB II, obtained from the National Bureau of Standards, enables the non-programmer to use a large digital computer to perform data, statistical and numerical analysis without having any prior knowledge of computers or computer programming.

```
jobname,CM170000,....      name / code
CHARGE,....
MSACCES,<password>.
MSFETCH,OMNITAB,UN=CSYS.
OMNITAB.
...
```

*** Pascal 6000 ***

Pascal is a modern programming language designed for general-purpose applications. It is derived from Algol-60. The Pascal 6000 Release 3 compiler was obtained from the University of Minnesota. Also obtained were several Pascal tools for use with Pascal source programs.

References: "Pascal - User Manual and Report", Jensen and Wirth.

Machine-readable documents:

```
BEGIN,DOCGET,,PASCAL,,rtn,OUTPUT,MSACCES=<pw>.
```

```
where rtn is PASCAL - the compiler      (92 pages)
           PASCLIB - the object library (77 pages)
           PTOOLS  - the tools package  (34 pages)
```

*** PL/I ***

PL/I is a block structured language. The CDC PLI compiler is a subset of ANSI PL/I.

Usage: PLI,<keyword-params>. comments

PL/I programs may call FORTRAN subprograms or many built-in functions from PLILIB.

*** Sharp Data Base Management System ***

SHARP, Ships Analysis and Retrieval Program, is a generalized Data Base Management System (DMS) developed at the Naval Ship Research and Development Center (DTNSRDC), Carderock, Maryland. SHARP is on the CDC CYBER 750 computer under NOS/BE and Intercom, which provides time-sharing access to the system.

SHARP is a self-contained DMS, designed to allow non-technical persons to define, build, maintain, and interrogate data bases without requiring application program interfaces. SHARP allows on-line and off-line access to data bases. User-oriented, English-like language is used for both retrieval and report generation and is highly suited for interactive use from remote terminals. Application program interfaces are available if needed for user programs to access data in the data base and for generating special reports on data retrieved during a SHARP interrogation of the data base.

Current data base applications under the SHARP DMS include:

- . technical document (bibliographic)
- . ship overhaul experience
- . ship operating experience
- . Computer Center expense accounting
- . Library circulation
- . system software accounting
- . Ocean Science Research Project information

The major features of SHARP include:

- . generalized data base definition
- . generalized user-oriented format check capability
- . partially inverted file structure
- . variable length and variable data record structure
- . ability to change data base definition without rebuilding data base
- . on-line file maintenance with free form direct entry capability
- . English-like, user-oriented language for query commands report definitions, and format checking
- . interactive report definition with ability to catalog defined reports for future selection
- . on-line interactive and batch retrieval
- . on-line ad hoc computation capability
- . host language interface
- . multi-level sort capability

SHARP was developed in COBOL to facilitate modification. In addition to the CDC 6000/CYBER, SHARP is operational on the Data General Eclipse mini computer and the CDC CYBER 173.

Reference: SHARP Data Base Management System User Information
Manual TM-188-76-1

Assistance: Code 1822, (202) 227-1533

*** Simscript II.5 ***

Simscript II.5 is a general purpose simulation language, with features such as recursion and dynamic storage allocation. It is also a discrete event simulator, with features such as flexible modelling of physical objects and automatic collection of statistics.

```
ATTACH,SIMII5.          <-- compiler
SIMII5,<options>.
ATTACH,SIM2LIB.         <-- object library
LGO.
```

References: "Simscript II.5 Programming Language", P. J. Kiviat, R. Villanueva, H. M. Markowitz, 1975.

"Simscript II.5 Reference Handbook", 1976.

"Simscript II.5 User Information Manual", CDC publication number 84000460

*** Snobol ***

Snobol is a string manipulation interpretive programming language designed to operate upon character strings rather than numerical data. Snobol does accept integer and real numeric operations, however execution time is greater than for a compiler such as FORTRAN. Snobol provides an extensive list of functions to operate upon strings. Patterns used for substring matching may be simple or complicated. Snobol is a public-access permanent file.

Reference: "The Snobol4 Programming Language", R. E. Griswold

A 37-page document on how to use Snobol may be printed by:

```
BEGIN,DOCGET,,OTHER,,SNOBOL,OUTPUT,MSACCES=<pw>.
```

Snobol is not supported by CDC or the Computer Center. Requests for credit because of software failure will not be honored.

*** Sort/Merge 4 ***

Sort/Merge version 4 is a flexible system routine for sequencing and combining files. In addition to the usual alphanumeric fields, sort may be on floating-point, fixed-point (integer) and logical type fields. A user option allows change of sort sequence.

The Sort/Merge system may be called directly as a stand-alone system with control cards to establish files and sort fields. In COBOL, the sort verb uses the system as a subroutine. FORTRAN has a set of subroutines which may be used to call Sort/Merge within an FTN program.

Guidelines for use of control card version of Sort/Merge:

1. The SORTMRG control statement has options for rewinding the directive and list files. Rewind all other files concerned in the sort (unless special positioning of an existing file is needed) before sorting.

2. Sort/Merge uses Record Manager as its input/output processor. FILE commands are required for all files to be sorted or merged and for the resulting output file.

3. INPUT is a valid file for sort input. The special system files INPUT, OUTPUT and PUNCH do not require FILE commands. They always have default BT=C and RT=Z.

4. OUTPUT is a valid file for sort output. The first character of each record is used for carriage control.

5. If no input FILE statement is supplied, the user must specify an EXIT 1 on the "owncode" directive and provide a routine to read the input records.

6. If Sort/Merge directive are not in the input stream, but are already stored as unit records on file "ABC", use 'SORTMRG(I=ABC)' or 'SORTMRG(I=ABC/R)' to indicate the source of the sort directives. (The second form will rewind "ABC" before reading the directives.) If the directives are on the UPDATE "COMPILE" file, use 'SORTMRG(I)' or 'SORTMRG(I/R)'.

7. Job card CM should be at least 12000 octal more than is needed to run the COBOL or FORTRAN program which uses Sort/Merge internally.

8. Sort/Merge 3.0 directives are accepted if the parameter "6C" is added to the SORTMRG command (i.e., SORTMRG(6C)). FILE commands are not required with Sort/Merge 3.0 control cards. The version 3.0 RECORD directives are converted internally into corresponding Record Manager FILE commands.

9. Records with identical sort keys are sequenced arbitrarily unless the "RETAIN" parameter is added to the "OPTIONS" directive. When "RETAIN" is specified, identical keys are sequenced in the order in which they were read.

*** Sample Sort/Merge 4 Setups ***

1. Sort card image tape on endpunching (columns 73-80):

```
xxxxST,MT1.                                name / code
CHARGE,....
LABEL,CARDS,L=xxxxCARDS,D=HY,R,VSN=CA7777,NORING.
REQUEST,SCRATCH,*PF.                      <-- scratch file on disk
FILE(CARDS,MRL=80,BT=C,RT=Z)
FILE(SCRATCH,MRL=80,BT=C,RT=Z)
SORTMRG.
COPYSF(SCRATCH,OUTPUT)                    <-- list sorted file
<eor>
SORT
FILE,INPUT=CARDS(CU),OUTPUT=SCRATCH
FIELD,ENDPCH(73,8,DISPLAY)
KEY,ENDPCH(A,COBOL6)
END
<eoi>
```

2. Create and sort unit records. Records are sorted on 23-25 (ascending) and 2-16 (descending). Note: sorting could have been done within the FTN program.

```
jobname,....                                name / code
CHARGE,....
FTN.
LGO.
FILE(TAPE7,MRL=140,BT=C,RT=Z)
FILE(TAPE9,MRL=140,BT=C,RT=Z)
SORTMRG.
  (other control cards to process sorted file TAPE9)
<eor>
  (FORTRAN source creating TAPE7 on disk with formatted writes)
<eor>
  (FORTRAN data)
<eor>
SORT
FILE,INPUT=TAPE7,OUTPUT=TAPE9
FIELD,MAJOR(23,3,DISPLAY),MINOR(2,15,DISPLAY)
KEY,MAJOR(A,COBOL6),MINOR(D,COBOL6)
END
<eoi>
```

3. Sort using Sort/Merge 3.0 directives. Sort I/O files have 40-word records with 12 records per block.

```
jobname,PE2,....          name / code
CHARGE,....
LABEL,OLD,D=PE,L=SORTIN,VSN=CA9999,NORING.
LABEL,NEW,D=PE,L=SORTOUT,VSN=CA9998,RING.
FILE(OLD,BT=K,RT=F,MRL=400,RB=12,MBL=4800)
FILE(NEW,BT=K,RT=F,MRL=400,RB=12,MBL=4800)
LDSET(FILE=NEW/OLD)
SORTMRG(6C)
<eor>
      (Sort/Merge 3.0 directives)
<eoi>
```

*** Sort/Merge 5 ***

Sort/Merge version 5 is a flexible system routine for sequencing and combining files. In addition to the usual character fields, sort may be on floating-point, fixed-point (integer) and logical type fields. A user option allows change of sort sequence. *

The Sort/Merge system may be called directly as a stand-alone system with optional control cards to establish files and sort fields. FORTRAN 77 (FTN5) and COBOL 74 (COBOL5) use a set of subroutines to access Sort/Merge. The COBOL5 control statement directs the SORT verb to use Sort/Merge 5 or 4. Sort/Merge 5 may also be used interactively.

** Using Sort/Merge 5 Without Separate Directives **

Sort/Merge 5 may be executed with a single control statement (which may be continued) and no separate file of directives:

```
SORT5.<parameters>
MERGE.<parameters>
```

The <parameters> are either keyword or positional or both. Each line to be continued must end with 2 or more periods (the range parameter may not be split) and starting the continuation line with one period. the SORT5 or MERGE statement may be up to 240 characters, not counting continuation characters. When this form is used interactively, it is limited to one card image (80 columns). If continuations are required, use a separate directive file or a CCL procedure.

** Using Sort/Merge 5 With Separate Directives **

Sort/Merge 5 may be executed with the directives in a separate file:

```
SORT5.dir=<directivefile>
MERGE.dir=<directivefile>
```

It may also be called with some parameters in the execute statement and others in a separate file:

```
SORT5.<params>,dir=<directivefile>,<more params>
MERGE.<params>,dir=<directivefile>,<more params>
```

* - Until NOS/BE is upgraded beyond level 552, use the following to access Sort/Merge 5:
ATTACH,SRT5LIB564,ID=CSYS.
LIBRARY,SRT5LIB.

** Interactive Sort/Merge 5 **

Sort/Merge 5 may be executed interactively:

SORT5.DIALOG=YES
MERGE.DIALOG=YES

The system will guide the user through the definition of all required parameters. Local files INPUT and OUTPUT are connected automatically.

** Sort/Merge 5 Guidelines **

1. The input, output and directive files (except files INPUT and OUTPUT) are rewound before and after use. To inhibit rewind, use separate FILE commands before invoking Sort/Merge. (see page 5-11: CF, OF).
2. Sort/Merge assumes all files have BT=C and RT=Z (default CDC sequential files) with a maximum record length (MRL) of 150 characters. To change any of these, use separate FILE commands before invoking Sort/Merge. Note that if MRL is very small, using separate FILE commands should speed up the sort.
3. If the files have BT=C, RT=F or BT=I, RT=W, and you are sure of the integrity of the files, the FASTIO option may be used to bypass record manager, however, not all errors may be detected.
4. INPUT is a valid file for Sort/Merge input.
5. OUTPUT is a valid file for Sort/Merge output. The first character of each record is used for carriage control.
6. If 'FROM=\$NULL' is specified, the user must provide exit 1 (and perhaps exit 2) routines and specify 'OWN1=<exit1name>' (and 'OWN2=<exit2name>') and may require OWNFL, OWNMRL, and OWNT parameters.
7. If 'TO=\$NULL' is specified, the user must provide exit 3 (and perhaps exit 4) routines and specify 'OWN3=<exit3name>' (and 'OWN4=<exit4name>') and may require OWNFL, OWNMRL, and OWNT parameters.
8. Input and directive files are not read past any embedded EOR or EOF.

9. Sort/Merge 5 requires a minimum of CM30000. The default is CM60000.
10. Sort/Merge 5 and Sort/Merge 4 are functionally equivalent, but version 4 directives may not be used with version 5. However, the 'OWNT=OLD' parameter allows owncode subroutines with the version 4 calling sequence to be used by version 5.
11. Records with identical sort keys are sequenced arbitrarily unless the 'RETAIN=YES' parameter is specified, in which case they are sequenced in the order in which they are read.
12. The STATUS parameter allows an error level code to be passed to one of the CCL registers (R1, R2, R3, R1G, EF, EFG). It can then be tested in a CCL procedure. If STATUS is specified, Sort/Merge will not abort on any error condition.
13. The SUM parameter can be used to combine records having equal keys into a single record with one or more fields summed.
14. COBOL5 programs may use Sort/Merge 5 via the SORT verb (COBOL5, ..., SORT5, ...) or direct calls ('ENTER FTN5 SM5xxx').

*** Sample Sort/Merge 5 Setups ***

1. Sort card image tape on endpunching (columns 73-80).

```
jobname,CM60000,GE1.                name / code
CHARGE,....
LABEL,CARDS,L=xxxxCARDS,D=GE,R,VSN=CA7777,NORING.
SORT5.FROM=CARDS,TO=DISK,KEY=((73..80,COBOL6,A))
UNLOAD,CARDS.
COPYSF,DISK,OUTPUT.                  <-- list sorted file
<eoi>
```

2. Create and sort unit records. Records are sorted on 23-25 (ascending, written with FORTRAN I3 format) and 2-16 (descending). Note: sorting could have been done within the FTN5 program.

```
jobname,....                name / code
CHARGE,....
FTN5,OPT=1,R=2.
LGO.
SORT5.TAPE7,TAPE9,((23,3,NUMERIC_FS),...
.(2..16,COBOL6,D))
  (other control cards to process sorted file TAPE9)
<eor>
  (FORTRAN source creating TAPE7 on disk with formatted writes)
<eor>
  (FORTRAN data)
<eoi>
```

3. Sort BT=C, RT=Z records in file OLD. Output goes to file NEW. The sort key is the entire record. Records are <= 150 characters.

```
jobname,....                name / code
CHARGE,....
  (control statements to attach or create file: OLD)
SORT5.
  (control statements to process the sorted file: NEW)
<eoi>
```

4. Merge 3 presorted files. The files have BT=I, RT=W (FORTRAN unformatted writes). Equal-keyed records are to be combined and have the real numbers in words 4-5 (31-40, 41-50) and the integer in word 10 (91-100) summed individually. The file is to be sorted ascending on text in columns 1-30 and descending on real numbers in word 7 (61-70). The directives are in the input job stream.

```
jobname,CM110000,....          name / code
CHARGE,....
    (control statements to attach or create the 3 files: TAPE1,
      TAPE2, TAPE3)
MERGE.DIR=INPUT.
    (control statements to process the sorted file: TAPE4)
<eor>
MERGE, FROM=(TAPE1,TAPE2,TAPE3)
MERGE, TO=TAPE4
MERGE, KEY=((1..30,COBOL6,A),(61,10,REAL,D))
MERGE, ENR=20000                <-- estimated number of records
MERGE, RETAIN
MERGE, SUM=((31..40,REAL,2),(91..100,INTEGER))
MERGE, FASTIO
<eoi>
```

*** SPSS (proprietary) ***

Statistical Package for the Social Sciences (SPSS) is an open-ended integrated system of statistical programs embedded in a single control program. The CDC version was obtained from Northwestern University. SPSS is a batch system written mostly in FORTRAN. This package (version 9.0) is more versatile than the BIMEDP routines (page 19-5), since many different statistics can be performed on the same data in one run. At least CM140000 should be specified on the job card; however, the system will reduce memory to the lowest needed for each type of statistic.

Current programs include:

- . Descriptive statistics
- . Cross tabulation
- . Pearson and rank order correlation
- . Partial correlation
- . Multiple regression analysis
- . Nonlinear regression analysis
- . Guttman scale analysis
- . Joreskog factor analysis
- . Canonical correlation
- . Bivariate printer plotting
- . T-test
- . One-way analysis of variance
- . N-way analysis of variance
- . Multivariate analysis of variance and covariance
- . Discriminant analysis
- . Nonparametric statistical tests
- . Item and scale reliability analysis
- . Tetrachoric correlations
- . General and three-stage least squares
- . Spectral analysis of time series
- . Survival analysis
- . Multiple response frequency and cross tabulation
- . Box-Jenkins (new in version 9)

References: "SPSS, Combined Edition", Nie, Hull, McGraw-Hill, Inc, 1981
(ISBN07-079052-3) (includes 2nd Ed. and 7-9 update)
"SPSS Primer", Klecka, Nie, Hull, McGraw-Hill, Inc., 1975
Note: the XSPSS books do not describe our system.

The following documents may be printed:

BEGIN,DOCGET,,SPSS,,doc,OUTPUT,MSACCES=<pw>.

where doc is SPSSGEN - document 187 (15 pages) general description
SPSSV90 - document 457B (93 pages) CDC update
VER90 - document 86 (7-page summary)
XREF90 - document 508 (6 pages) (cross reference to
other SPSS documentation)
ERRPT90 - summary of reported problems (48 pages)
JFACTOR - document 412
MANOVA - document 588 (91 pages)
SUMTABL - document 411

** SPSS Examples **

1. Skeleton example:

```
xxxxyyy,CM140000.          name/code
CHARGE,....
ATTACH,SPSS.
SPSS(<param>)    <-- <param> - optional parameters as described
<eor>            in Appendix F of SPSS
...
<eoi>
```

2. Control statements in file other than INPUT:

```
ATTACH,1fn1,....
SPSS(I=1fn1)
```

3. Raw input data in file other than INPUT (control card "INPUT MEDIUM DISK", no data follows "READ INPUT DATA"):

```
ATTACH,1fn2,....
SPSS(D=1fn2)
```

If raw input data is on fixed blocked stranger tape, use RL=nnn,BL=xxx (SPSS,D=1fn2,RL=nnn,BL=xxx.) to indicate size of record and block.

4. To create a save file (control card "SAVE FILE"):

```
REQUEST,1fn3,*PF.
SPSS(S=1fn3)
CATALOG,1fn3,....
```

5. To get a previously saved system file (control card "GET FILE ..."):

```
ATTACH,1fn3,....
SPSS(G=1fn3)
```

6. If only one procedure is to be run, use X=0 on the SPSS statement to avoid writing binary file XXSPS2, saving time and disk space.

*** System 2000 (proprietary) ***

System 2000, version 3.0, is a Data Base Management System available to all Navy users of the DTNSRDC CYBER computers. System 2000 has a self-contained language module which is suitable for interactive use. A report writer feature is available as an extension to the self-contained language use. System 2000 also supports FTN4, FTN5, COBOL 4, and COBOL 5 access to data bases via programming language extension (PLEX) statements.

Note

System 2000, Version 2.80, is still a supported version of System 2000. The 2.80 procedure for calling System 2000 may be accessed by replacing S2K30 with S2K280 in the following examples. Version 2.80 does not support Fortran 5. For further comments on the differences between Version 2.80 and Version 3.0, see the System 2000 Newsletter, publication number 221175.

** Approximate Field Lengths **

self-contained languages	57000B
report writer	66000B
PLEX programs	small version 50000B*
	medium version 57000B*
* plus program and object routines	

** Access to S2000 **

System 2000 must be accessed via a standard procedure set up:

Example 1. To get into S2000 self-contained languages:

```
BEGIN,S2K30.  
S2K30.
```

Example 2. Compile and execute COBOL 5 PLEX program.

```
BEGIN,S2K30,,COB.  
PLICOB,...  
COPYR,PLILGO,LGO,2.  
COBOL5,I=TAPE3.  
COPYF,PLILGO,LGO.  
LGO.
```

Example 3. Compile and execute a COBOL 5 PLEX program with the small version of S2K30.

```
BEGIN,S2K30,,COB,SMALL.  
PLICOB,...  
COPYR,PLILGO,LGO,2.  
COBOL5,I=TAPE3.  
COPYF,PLILGO,LGO.  
LGO.
```

Example 4. Compile and execute a COBOL 4 PLEX program.

```
BEGIN,S2K30,,COB.  <-- for small version, add "SMALL"  
PLICOB,...  
COPYR,PLILGO,LGO,2.  
COBOL,I=TAPE3.  
COPYF,PLILGO,LGO.  
LGO.
```

Example 5. Compile and execute FORTRAN PLEX program.

```
BEGIN,S2K30,,FOR.  <-- for small version, add "SMALL"  
PLIFOR,...  
COPYR,PLILGO,LGO,2.  
FTN5,I=TAPE3.      <-- or FTN4,I=TAPE3.  
COPYF,PLILGO,LGO.  
LGO.
```

Example 6. Create a report writer user exit.

```
BEGIN,S2K30,,RWX.  
COPYR,RWEXIT,LGO,3.  
FTN5,... or FTN4,... or COBOL,... or COBOL5,....  
LGO.
```

** General Notes **

1. When executing a PLEX absolute program, you must still do a BEGIN,S2K30.
2. When executing an absolute COBOL 4 PLEX program which uses segmentation, the absolute program must be attached with local file name COBCODE.

** Local Requirement **

DTNSRDC requires that the user ID (xxxx) be specifically declared within all S2000 programs. This is done as follows:

ID IS xxxx: self-contained language format

CALL SETIDX (yyy) FORTRAN PLEX statement
where yyy = "xxxx"

CALL SETIDX USING yyy COBOL PLEX statement
where yyy is defined as
77 yyy PIC X(10) VALUE "xxxx".

Use procedure S2KRNM (see CLIB/P) to rename account number on cataloged S2000 data base files.

** Documentation **

Documentation which may be obtained from the S2000 vendor (INTEL Corporation) or ordered from User Services:

	publ. no.
System 2000 version 3.0 Newsletter	221175
Messages and Codes for CDC release 3.0	1090
Define - Language Specification Manual	1010
COBOL PLEX - Language Specification manual	1020
FORTTRAN PLEX - Language Specification manual	1022
QUEST - Language Specification Manual	221126
Report Writer	221171
LSM/ADM	1013

Documentation available on the procedure for calling System 2000, Version 3.0:

BEGIN,DOCGET,,PROCFIL,,S2K30,MSACCES=<pw>.

** SAVE a Data Base for Backup **

SAVE on tape: jobname,PE1,...
 CHARGE,...
 VSN,TAPE999=<storage id>.
 LABEL,TAPE999,L=<label id>,W,D=PE,RING. <-- must be
 TAPE999

 BEGIN,S2K30.
 S2K30,TP.
 <eor>
 USER,<user password>: ID IS <user id>:
 DBN IS <data base name>:
 CONTROL:
 SAVE DATA BASE ON TAPE999:
 EXIT:
 <eoi>

SAVE on disk: jobname,....
 CHARGE,...
 REQUEST,TAPE999,*PF.
 BEGIN,S2K30.
 S2K30,TP.
 CATALOG,TAPE999,...
 <eor>
 (same as above)

SAVE on MSS: jobname.
 CHARGE,...
 BEGIN,S2K30.
 S2K30,TP.
 MSACCES,<password>.
 MSSTORE,TAPE999,<mfn>.
 <eor>
 (same as above)

Notes on SAVE:

1. The use of TP parameter is to force S2000 to write on TAPE999, thus allowing the use of a labelled tape or a disk file. Without the TP parameter, S2000 will issue a request using the reel number specified in "SAVE DATA BASE ON <file id>".
2. The SAVE to tape must be a batch job (no interactive tape usage is permitted), but the save to disk may be done interactively.
3. The data base is not affected by the SAVE. It is not released unless the user explicitly does a "CONTROL: RELEASE:" sequence.
4. 'SAVE' tapes written by S2000 are SI tapes (BT=C, RT=Z).

**** RESTORE a Data Base from Backup ****

```
RESTORE from tape:  jobname,PE1,...
                   CHARGE,...
                   VSN,TAPE999=<storage id>.
                   LABEL,TAPE999,L=<label id>,R,D=PE,NORING.
                   BEGIN,S2K30.
                   S2K30,TP.
                   <eor>
                   USER,<user password>: ID IS <user id>:
                   RESTORE <data base name> FROM TAPE999:
                   EXIT:
                   <eoi>
```

Notes on RESTORE:

1. If backup is on a permanent file, use
ATTACH,TAPE999,....
in place of VSN and LABEL statements.
2. If backup is on the Mass Storage System, use
MSACCES,<password>.
MSFETCH,TAPE999,<mfn>.
in place of VSN and LABEL statements.
3. The data base must not already exist at the time RESTORE is performed. That is, it must have been removed via the S2000 "RELEASE" command.
(See 4 for exception).
4. The data base may be renamed during RESTORE. This option allows the old data base with the original name to co-exist with the restored database having a new name.
RESTORE <old name> = <new name> FROM TAPE999:
5. The user ID may be changed during restore simply by using the "ID IS <user id>:" directive to supply the new ID.

*** System 2000, Version 2.60 (proprietary) ***

System 2000, version 2.60, is an obsolete version of System 2000. All new applications should use Version 3.0. A report writer feature is available as an extension to basic natural language use. S2K260 also supports FTN4 or COBOL 4 access to data bases via programming language interface (PLI) statements. Version 2.60 will be removed from the system in March 1985.

** Approximate Field Lengths **

basic natural language	53000B
report writer	60000B
PLI programs	53000B *

* plus program and object routines

** Access to S2K260 **

System 2000, Version 2.60, must be accessed via a standard procedure set up:

Example. To get into S2000 natural language:

```
BEGIN,S2K260.  
S2K260.
```

Example. Compile and execute COBOL 4 PLI program.

```
BEGIN,S2K260,,COB.  
PLICOB,....  
COPYR,PLILGO,LGO,2.  
COBOL,I=TAPE3.  
FTN4,I=TAPE3.  
COPYF,PLILGO,LGO.  
LGO.
```

Example. Compile and execute FORTRAN V4 PLI program.

```
BEGIN,S2K260,,FOR.  
PLIFOR,....  
COPYR,PLILGO,LGO,2.  
FTN4,I=TAPE3.  
COPYF,PLILGO,LGO.  
LGO.
```

** S2K260 Documentation **

S2000 Reference Manual
Basic S2000 Natural Language
Immediate Access
Report Writer
CDC Machine Dependent Information
S2000 Users Manual
COBOL & FORTRAN Procedural Language Interfaces
S2000 System Support Manual
System 2000 version 2.60 Newsletter (April 1977)

Documentation available on the procedure for calling System 2000,
Version 2.60:

BEGIN,DOCGET,,PROCFIL,,S2K260,MSACCES=<pw>.

*** Text Processors ***

Two text processors (RNF and PROSE) are available on the CDC computers at DTNSRDC. Both are derived from the RUNOFF text processor and provide line filling, pagination, left/right margin justification, chaptering, sectioning, etc.

** RNF **

In addition, RNF also provides numbered lists and sublists, a macro facility, etc.

RNF was obtained from the University of Illinois by one of our former customers (CERL).

A document (43 pages + cover) describing RNF may be printed by:

BEGIN,DOCGET,,OTHER,,RNF,OUTPUT,MSACCES=<pw>.

** PROSE **

In addition, PROSE also provides discretionary hyphenation, automatic sorted indexing, etc.

An associated program, WRITE, is available to convert PROSE output to an appropriate output device (line printer/interactive terminal) and character set (mixed-case/upper-case-only).

PROSE and WRITE were obtained from the University of Minnesota.

Documents describing PROSE (30 pages + cover) and WRITE (9 pages + cover) may be printed by:

BEGIN,DOCGET,,OTHER,,PROSE,OUTPUT,MSACCES=<pw>.

BEGIN,DOCGET,,OTHER,,WRITE,OUTPUT,MSACCES=<pw>.

***** USER HELPS *****

*** Conversion Aids ***

** Non-standard FORTRAN Usage **

The ANSI flag of the FTN5 or FTN4 compiler on the CDC CYBER will indicate non-standard FORTRAN usage. Use FTN5,ANSI=T,.... or FTN4,EL=A,....

** Convert Double Precision to Single Precision **

CVT360 is a Snobol program to convert double precision IBM or VAX FORTRAN programs to single precision CDC FORTRAN V4. See page 19-3.

** EBCDIC to BCD **

At Central Site or at a high-speed, remote batch terminal, put '29' into columns 79-80 of the end-of-record (7/8/9) card preceding the deck. Then use
COPYCF,INPUT,PUNCH.

** BCD to EBCDIC **

CV029 (which replaced CV29) is a procedure to convert from BCD to EBCDIC at a remote batch terminal. At Central Site, use
ROUTE,1fn,DC=PU,EC=029.

** COBOL 4 to COBOL 5 **

See page 14-8 for a discussion of the COBOL 4 to COBOL 5 Language Conversion System (LCS).

** FTN4 to FTN5 **

See page 13-19 for a discussion of the FORTRAN Extended, Version 4, to FORTRAN Extended, Version 5, language conversion program (F45).

See CONV manual for additional aids.

*** Communications With Operator ***

*** "PM" Carriage Control Disposition at Remote Sites ***

"PM" in columns 1 and 2 of an output print line will stop the line printer and display a message at the operator's console of medium-speed remote batch terminals or the teletype at a high-speed computer terminal. The message (columns 3-32 of the PM print line) should inform the operator which print forms are to be mounted and/or which carriage control tape is required before the operator restarts the printer from his console. (See page 5-15 for types of special forms.) Special forms usage at Central Site must be by the ROUTE command (see page 5-12).

Remote site special forms may be accomplished in one of four ways. It is the user's responsibility to ensure that the standard forms or carriage control tapes are restored to the printer at the end of the job. Be sure that the special forms you need are available at the terminal.

** Examples **

1. By control statement (using procedure 'PM'):

```
...  
...  
BEGIN,PM,,<fc>.      << message to mount required forms  
...  
    (execute program which creates the special form output)  
...  
EXIT,U.  
BEGIN,PM.            << message to restore printer  
<eor>  
...  
    (records as needed for intermediate steps)  
...  
<eoi>
```


2. By control statement (using data records in the job stream):

```
...  
...  
COPYCR(INPUT,OUTPUT)  
...  
    (execute program which creates the special form output)  
...  
EXIT,U.  
COPYCR(INPUT,OUTPUT)  
<eor>  
PM mount special forms      type  
<eor>  
...  
    (records as needed for intermediate steps)  
...  
<eor>  
PM remount standard forms  
<eoi>
```

3. Within program execution (using subroutine 'PM'):

FORTRAN 5 example

```
...  
CALL PM (-1, '<fc>')  
...  
CALL PM (-1, ' ' )  
...
```

4. Within program execution (using a FORMAT statement):

FORTRAN 5 example

```
...  
PRINT 10  
10 FORMAT ('PM mount special forms      type')  
...
```

*** Dayfile Message Display ***

To display a message at a control point and in the dayfile:

COBOL	DISPLAY
FTN5, FTN4	CALL REMARK or CALL DISPLA
control statement	COMMENT.

These messages will not flash or wait for operator action.

*** Reserved Words ***

Files which have special dispositions at end-of-job may not be used as file names on device sets or as overlay file names.

file name	may only be used as
INPUT	job stream or TTY input
OUTPUT	job stream or TTY output
PUNCH	BCD punch file
PUNCHB	binary punch file
P80C	special binary punch file
FILMPL	former SC-4020 SCORS plot file
FILMPR	former SC-4060 meta plot file
PLOT	former remote Calcomp plot file
PTAPE	ASCII paper tape output from APT

Interactive intercom will not allow most single letter names for object files since these are abbreviated intercom batch terminal commands.

Intercom users should avoid any local files of the same name as intercom commands, since many of these have a preset field length which could cause a binary program not to load.

CCL verbs should not be used as local file names (lfn's) or as program names in editlib libraries.

Local file names beginning with 'ZZ' are reserved to NOS/BE and should be avoided by the user.

*** Estimating Print Requirements ***

Because a print line normally occupies only as much disk space as is needed, and may have fast-feed carriage control, it is not possible to calculate the amount of paper needed to print an output file. However, a very rough estimate can be made.

If the output file has been cataloged, the CT messages or audit gives the number of PRUs (sectors).

Under Intercom, the FIND (J), MYQ and Q commands give the number of sectors / 1000B (512). These should be converted to decimal sectors.

Each sector holds 640 characters (including zero-byte terminators) or about 5 nearly-full lines. A page with 60 such lines uses about 12 sectors. Thus, the number of pages is approximately (sectors / 12).

Shorter lines increase the number of lines per sector; the use of fast-feed carriage control and/or short pages increases the number of pages for a given number of lines. At one extreme, a full page with only 5 characters per line needs only one (1) sector, and the number of pages is the number of sectors.

*** Generating End-of-Record Marks ***

The characters which will generate end-of-record or end-of-file marks depend on the program/mode being used.

<u>program/mode</u>	<u>end-of-record</u>	<u>end-of-file</u>
CCL	.EOR	.EOF
UPDATE	*WEOR *CWEOR	
NETED	EOR	EOF
EDITOR	*EOR	*EOF
input to a program	%EOR	%EOF

The 7/8/9 card (or end-of-record) has different meanings depending on the local file name (lfn). When encountered on file INPUT, a 7/8/9 (eor) is an EOR for the copy utilities, but an EOF for other programs. If INPUT is copied to another lfn, then the programs reading that lfn will treat them as EOR's. Thus, for example, if a card deck has several data decks separated by 7/8/9 cards, one program execution is required for each record. However, if copied to another lfn, a single program execution will read all data.

***** INTERACTIVE GRAPHICS *****

On the CDC computers at DTNSRDC, interactive graphics is supported by the Tektronix display terminals.

*** Tektronix 4015/4014 ***

The Tektronix 4015/4014 may be a synchronous or asynchronous computer display terminal and is capable of displaying both alphanumeric characters and graphic data. The display remains visible until it is erased and does not require continual refreshing. The display area is 14.5 by 10.9 inches in a 4096 by 3120 grid. Four character sizes are available. To change character size, put the LOCAL/LINE switch on LOCAL and type:

ESC key then	8	for maximum of 74 characters/line;	35 lines
	9	81	38
	:	121	58
	;	133	64

The : size is adequate for reading. The 4015/4014 displays two columns of information (the second column is not full width).

When no data is input for 30 seconds, the screen will dim to protect the phosphor. Depress the shift key to brighten the screen again.

To operate as an interactive terminal to CDC computers, a 300-baud, 1200-baud, or 4800-baud ASCII modem is required. For a 1200-baud modem, the HS button on the modem must be pushed in.

- . Turn 4015 terminal switch and hard copier on. Set LIGHT/DARK dial on hard copy unit for proper density.
 - . Set LOCAL/LINE switch to LOCAL.
 - . RESET PAGE (more than once if the screen does not clear), then depress the SHIFT and RESET PAGE keys together.
- While terminal and hard-copy unit (if any) warm up, check:

	Synchronous	Asynchronous
. ASCII/BCD	ASCII	ASCII
. rotary baud	external for 4800	
(rear of terminal)	300 or 1200	300 or 1200
. clear write	off	off
. code expander	on	off
. set half duplex	no	simultaneous
		CTRL/SHIFT/P *
. select char size	ESC then :	ESC then :

- . set LOCAL/LINE switch to LINE.
- . Dial proper phone for modem speed (see pages 1-3,4).
- . When computer data tone answers, push DATA button and hang up.
- . Log in and run job.
- . At end of run, in command mode, type LOGOUT.
- . Disconnect the phone line.

* - do this only once.

Most DTNSRDC 4015 keyboards have the APL character set on top of the keys, so the proper CDC character set is the small characters on the front of several of the special symbol keys. The 4014 keyboard has the CDC character set.

When used as a synchronous terminal, the signals for several control functions differ from most interactive terminals:

Action	Synchronous operation	Asynchronous operation
-----	-----	-----
abort current command	BREAK	ESC % A
backspace a character	RUBOUT	BACKSPACE or CTRL-H
cancel a line	CTRL-U	CTRL-X

When used as an interactive typewriter terminal, two columns of commands will be accumulated on the screen unless the user depresses the RESET PAGE key. The user must RESET PAGE to allow next screenful of data to appear when listing output. Before using graphing routines at 4800 baud, enter a "SCREEN(80,66)" command.

Both the PLOT10 package and DISSPLA postprocessor have different names for synchronous and asynchronous. It is less efficient to use the 'CODE EXPANDER ON' (synchronous) with 300 or 1200 baud lines.

	synchronous	asynchronous
-----	-----	-----
. PLOT10 file name	TEK48	TEK30
. DISSPLA postprocessor	TEK480	TEK300

The PLOT10 subroutine RECOVER has been renamed to RECOVX; subroutine RESET may also be called as RESETT.

The first two calls to PLOT10 are:

```
CALL INITT (icps)           icps is 30 or 120
CALL TERM (3, 4096)
```

The recommended last call to PLOT10 is:

```
CALL FINITT (0, 500)
```

References:

PLOT-10 Terminal Control System Users Manual	062-1474-00
PLOT-10 Advanced Graphing II Users Manual	062-1530-00

*** Tektronix 4662 Plotter/Digitizer ***

The Tektronix 4662 plotter/digitizer may be used on a 300-baud line with terminals similar to Execuport 300, Hazeltine 1000 (RS-232-C interface) as a pen plotter using the Tektronix PLOT 10 package, or as an alphameric (95-character ASCII) hard copy device. It may also be used as a hard copy device on a Tektronix 4051.

Several software packages may be used with the 4662. The PLOT 10 package is library "TEK30" containing 200 subroutines for use with Tektronix 401x storage tube terminals. Special subroutines such as "PLON" and "PLOFF", digitizer call button and other software for the special features of the 4662 are in user library "TEK4662". Reference is "4662A01 PLOT 10 Utility Routines for Use With the 4662 Plotter User Manual". Most programs written for 11-inch Calcomp drum plotters may be output, with no program modification, to Tektronix terminals by suitable library replacement. Because there are duplicate entry points in the associated libraries, the LDSET must follow the specified order of libraries.

```
...  
ATTACH,TEK30.  
ATTACH,TEK4662.  
ATTACH,CALC936.  
ATTACH,CALCFN.  
LDSET,LIB=TEK4662/TEK30/CALC936/CALCFN.
```

The plotter is turned on by rocking the POWER switch to the right. Be sure to remove the protective cap from the small felt tip pen. The 4662 control switches are normally set to "0221" (terminal mute, RS-232 interface, device a, 300 baud). Depress LOAD, place one sheet of 11 by 17 inch paper on the platen and smooth it flat. Press LOAD again to release button and build up electro-static charge. When LOAD button is down, no output on the terminal SCOPE is transmitted to the plotter.

Hardware page scale margins may be set using the joy stick when only the LOCAL button is depressed. First position to the desired lower left corner and press the 'SET LOWER LEFT' button until it beeps (1 second), then position to the desired upper right corner and 'set' it similarly. Then check the settings of the currently defined page by pressing in sequence the LOCATE buttons. When used in the plot mode, the margins may also be set by software in the PLOT-10 package. In LOCAL mode the keyboard of the attached terminal inputs only to the plotter, not over the line to the computer.

To use the plotter as a hard copy alphameric device for the interactive terminal, the LOCAL and LOAD buttons must both be up. Size of printed letters is scaled automatically by the margin settings to allow 85 characters per line and 35 lines. Unless the SCREEN command is used, computer output lines will not exceed 72 characters.

The plotter may be used as a GIN (graphic input) digitizer device by using a special cross hair magnifier instead of the ball point pen.

*** Tektronix 4051 ***

The Tektronix 4051 is capable of being used in terminal mode as an asynchronous interactive computer display terminal with both alphanumeric characters and graphic data (similar to Tektronix 4012). The screen accommodates 35 lines of 72 characters or a grid of 1024 by 780 "tekpoints". Plot resolution is about 130 by 100. When no data is input for 2 minutes, the screen will dim to save energy. Depress the shift key to brighten the screen again.

The 4051 also has stand alone (off-line) capability in basic language and has a cassette of graphic off-line software called PLOT50. The power up (default) mode is the basic I/O mode.

To prepare the terminal for access to CDC computers, several commands must be entered before dialing the central computer.

Press "home page" key to clear and darken the screen
type in

CALL "CMINIT" where the " (quote) characters are typed in
CALL "TERMIN"

When used as in interactive typewriter terminal, two columns of commands will be accumulated on the screen. Then the cursor (which shows the next character position) returns to the top of the screen and overwrites characters. The "home page" key will clear the screen.

Plotting routines of the PLOT-10 package are in file "TEK30".

*** Tektronix 4027 Color Terminal ***

The Tektronix 4027 is an asynchronous color graphics terminal capable of displaying both alphanumeric characters and graphic data. The 4027 will produce colored vectors, display colored polygons and characters and will create and display colored symbols. A maximum of eight colors may be displayed at one time.

The 4027 operates in three modes: off-line local, interactive with PLOT-10 easy graphing software, interactive with black and white. Operating under any mode, the terminal screen can be divided into two different work areas. The bottom part of the screen displays the monitor area while the top part displays the workspace. Text typed into the monitor area is sent to the host computer, text typed into the workspace is stored there, and is not automatically sent to the computer. The workspace is used to display graphic data or special forms and for editing. Final edited text in the workspace can be sent to the computer. Use Polaroid 45 or SX-70 for color hard copy.

To operate Tektronix 4027 as an interactive terminal to the CDC computers, a 300- or 1200-baud ASCII modem is required. For a 1200-baud modem, the HS button on the modem must be pushed in.

- . Turn 4027 terminal switch on.
- . Type !SYS to display terminal setting table.
- . Transmitting baud rate (TB) should be set to 1200 or 300.
(!BAU 1200 or !BAU 300)
- . Receiving baud rate (RB) should be set to 1200 or 300. (Both transmitting and receiving baud should be the same.)
- . Duplex setting (DU) should be full (F).
- . Echo mode setting (EC) should be local (L). (!ECH L) See the Tektronix 4027 Operators Manual for other setting definitions. All others should be constant.
- . Dial the proper phone number for the modem speed (see page 1-3, 1-4).
- . When computer data tone can be heard, push the DATA button and hang up the receiver.
- . Log in.

** EZGR - Color Graphics **

- . Type 'ATTACH,EZGR' *** ("EZGR", which must be the local file name, is the PLOT-10 Easy Graphing software package)
- . Type 'EZGR' *** (execute program "EZGR")
- . The terminal will respond by clearing the screen, displaying a '-#' prompt with a bell ring. The cursor will appear at the bottom left hand side of the screen.
- . EZGR establishes a work area of 30 lines and a monitor area of four (4) lines.
- . Enter Easy Graphing commands.
- . External files may be attached prior to entering EZGR, but must be constructed in the format EZGR expects. Only five (5) files may be processed at a time.
- . A maximum of five (5) files can be saved within EZGR and cataloged under NOS/BE.
- . A maximum of five (5) files can be attached using the 'ATTACH' command under EZGR.
- . External files can be created using NETED or EDITOR and processed under EZGR.
- . All local file names must be a maximum of four (4) characters.
- . The BYE command will terminate the EZGR mode and return user back to the command mode of Intercom.

References: "PLOT-10 Easy Graphing for 4027 Color Graphics Terminal User's Manual"
"PLOT-10 Easy Graphing for 4027 Color Graphics Terminal User's Reference Card"

*** Interactive Data Display System (IDDS) ***

IDDS is an interactive graphics system used for data examination and manipulation on the CDC CYBER computer. IDDS capabilities allow the user to graph 2-D cross plotting, contouring, and 3-D perspective projection of any data that can be presented in 1-, 2-, or 3-dimensional arrays of numbers. This system can be accessed on any of the Tektronix terminals connected to the CDC CYBER. IDDS is driven by command words, so users do not have to write an interface program.

To learn how to use IDDS, do the following on a hard copy terminal:

```
BEGIN,IDDS
HELP,PRIMER                <-- introduction on how to
                             get started with IDDS
```

The system will respond with

```
PF CYCLE NO. = nnn @ PLEASE TYPE YOUR: USER INITIAL, NAME,
ADDRESS, PHONE, & "END"
```

User initials must be the same as LOGIN. This information is requested only the first time the user enters IDDS.

For assistance, contact:

Mary Beth Marquardt, Code 1843, (202) 227-1933

*** MOVIE.BYU ***

MOVIE.BYU is an interactive graphics package of FORTRAN programs capable of displaying 3-D mathematical, topological, or finite element models as line drawings or as continuous tone-shaded color images. MOVIE.BYU can be run on any Tektronix terminal including the Tektronix 4027 color graphics terminal. The only requirement for displaying data is that it be in the MOVIE.BYU data format. An interactive procedure is available to automate the conversion of a NASTRAN bulk data deck to the MOVIE.BYU data format.

Reference: Lipman, Robert R., "A Guide to MOVIE.BYU at DTNSRDC,"
Report DTNSRDC-84/007 (February 1984).

For assistance, contact: Robert Lipman, Code 1844, (202) 227-1660

*** NASTEK ***

NASTEK is an interactive computer graphics program that will display NASTRAN-generated plots on Tektronix 401x and 405x terminals. The output written by NASTRAN on the PLT2 file is used for input to NASTEK.

To run NASTEK, login at a Tektronix terminal and enter:

```
MSACCES,password
MSFETCH,NASTEK,UN=CARL
ATTACH,PLT2,pfn,ID=xxxx
NASTEK
```

supplying your MSS password, permanent file name and user ID.

Reference: Lipman, Robert R., "NASTEK-Interactive Display of NASTRAN-Generated Plots," Report DTNSRDC/CMLD-84/01 (January 1984).

For assistance contact: Robert Lipman, Code 1844, (202) 227-1660.

***** OTHER COMPUTER CENTER EQUIPMENT *****

This chapter describes other computers and computer-oriented hardware available.

*** Datagraphix Mini Autocom Microfiche System ***

The Mini Autocom system is controlled by a Lockheed minicomputer having (the maximum) 32K 16-bit word memory. It creates computer output microfiche (COM) from any print tape (CDC, IBM, Burroughs, etc.). Components of the system include:

- . KSR 743 electronic data terminal
(provides communication between operator and Miniform (the software which drives the Autocom))
- . Integrated read-only 9-track tape drive (6250/1600 bpi)
- . Two diskette drives
(providing external read/write storage facilities)
- . Full reversible microfiche system

The Mini Autocom generates a standard page size of 64 lines by 132 characters. It is equipped with a microfiche camera that creates government standard size (for computer printouts) microfiche recording ratio of 48x on 105mm film.

A special forms code (FC=WW) is available for automatic creation of microfiche output. A special program reformats user output and writes a (system) tape for later processing on the Mini Autocom. The user does not have to supply a tape for this. To use a previously-established COM program, its name must be specified in columns 2-6 of the first record of the file. This is then followed by the file to be put onto fiche. The file is ROUTED using:

ROUTE,out,DC=PR,TID=C,FC=WW,FID=fid.

where <fid> should be either <user initials> or *<user initials>.

Procedure COMQ may be used to create fiche output of data and source files, compilation and loader listings, and program output. It will create a file as described above and route it. COMQ allows for the creation of an eye-readable title. Any user-defined COM procedure which expects the eye-readable title information in columns 2-33 of the second record may use COMQ to access it. Standard procedures include:

COMP For compilation and load map listings.
 132 characters/line
 268 pages plus 2 index frames per fiche

COMPF Same as COMP, but with a box around each frame.

COMQ For regular output.
 132 characters per line
 270 pages (no index) per fiche

COMQF Same as COMQ, but with a box around each frame.

Other standard COM procedures, not available using COMQ, are listed below. When using one of these procedures, the COM request (available at the ADP Center) and the form describing titling and indexing (call User Services for assistance) must be used.

CSYS06 -- fixed blocked EBCDIC 9-track tapes, 1600 bpi
 132 characters/line, 64 lines/page max
 carriage control characters in print data

CSYS10 -- same as CSYS11, except no titling or indexing

CSYS11 -- CDC binary 9-track tapes, 1600 bpi
 procedure for compiler listings with subroutine names
 as index keys, titling optional

** Notes **

If the print data was paged for more than 64 lines/page (e.g., 8 lines/inch), it will overflow to the next frame.

A standard rectangular box frame is available.

Tapes may be labelled or unlabelled. Indicate which on the COM request form or OFLREQ (see page 23-3).

Wherever possible, binary tapes (COPYBF) should be used since more data can be placed on a tape in binary form. Coded tapes should be avoided (colons :) will not print and zero-byte terminators may print as colons).

*** Calcomp Model 936 ***

The CMLD Calcomp model 936 drum plotter is source code compatible with Calcomp 1051.

The following plotting papers and pens will be available to users:

1. 36.7 inch plain paper #500
2. 36.7 inch 10 divisions per inch (brown grid) #501
3. 15.4 inch plain paper #600
4. 15.4 inch 10 divisions per inch (brown grid) #601
5. 3 ball point pens
A call to subroutine NEWPEN (n), where n = 1, 2, or 3 will allow users to alternate 3 pens in one job. Colors green, red, blue and black are available. See Calcomp Basic Manual for description of NEWPEN.

The basic set-up for the Calcomp 936 will be as follows:

- * 15.4 inch paper (#600 or #601)
- * pen 1 (black)
- * pen 2 (blue)
- * pen 3 (red)

A change to wide paper (36.7 inch) or a change in pen arrangement will increase the basic charge rate.

The ability to display and search for block address is not available on the Calcomp 936. Users who need to select blocks from their Calcomp tape must use program GETCOP on library UTILITY (see CLIB/U).

*** Calcomp Plot Request ***

Plot Request (Form 10462/26) must be submitted with each tape. User must provide pertinent information for the first two sections from left including job order number, estimated plotting time to aid in scheduling of work, date, block numbers of reel if known (usually 0 to 999), reel number, expected number of plots, paper number and grid color from next page, pen type (ball point gives best results), user origin if nonstandard, user name, phone number.

The Plot Request may be generated interactively using procedure OFLREQ (see CLIB/P):

BEGIN,DOCGET,,PROCFIL,,OFLREQ,OUTPUT,MSACCES=<pw>.

*** Xerox 8700 Printer ***

The Xerox 8700 is a high-speed off-line laser printer at Central Site. CDC CYBER computer output can be printed full size or reduced in portrait style (8.5x11 vertical) or reduced in landscape style (11x8.5 horizontal). Output may be printed in duplex (two-sided) or simplex (one-sided). Several fonts and software-generated forms overlays are available. A variety of Xerox 8700 jobs have been developed for different styles, fonts, etc.

Output for the Xerox 8700 will be taken from the CDC computers every 30 minutes. The printing charge for most printouts is about the same as for Central Site on-line.

The Xerox 8700 can overprint in the same position. However, overprinting a character with itself will not produce a darker character.

One way to use the Xerox 8700 is via procedure XEROX. To obtain the document:

```
BEGIN,DOCGET,,PROCFIL,,XEROX,OUTPUT,MSACCES=<pw>.
```

This procedure executes program LASER to generate a data record ahead of the output to be printed on the Xerox. This data record tells the Xerox what job name and other parameters to use.

The shortest call: 'BEGIN,XEROX,,lfn,fid.' will print one copy of current contents of file <lfn> on standard landscape three-hole paper using duplex printing with no special forms. This is the usual Xerox output for full-line printouts such as compilations and formatted program output.

For multiple copies, give total # of copies:	COPIES=<#_of_copies>
If paper without holes is required:	PAPER=PLAIN
If simplex printing is necessary:	DUPLEX=NO
If a special form is necessary:	FORMS=<form_name>

For printing 1T-type output (documentation, procedures):

JOB=STDPRT	for pages up to 72 characters wide
JOB=ST4PRT or DOCPRT	87 characters wide

These use larger character fonts in portrait style similar to 1T printing.

Another way to use the Xerox 8700 is to execute the LASER program directly. Each call to LASER should be followed by some output. This is the only way to include the dayfile with the rest of the batch job output (see example 2). It can also be used to force a new page at any point in your printout. When your Xerox file is complete, route it to the Xerox output queue using an appropriate forms code in the ROUTE statement. A copy of the document describing LASER may be obtained by:

```
BEGIN,DOCGET,,OTHER,,LASER,OUTPUT,MSACCES=<pw>.
```

**** Examples ****

1. Print the document for NETED:

```
BEGIN,DOCGET,,OTHER,,NETED,OUT,MSACCES=<password>.
```

```
BEGIN,XEROX,,OUT,xxxx,JOB=DOCPRT.
```

-or-

```
BEGIN,DOCGET,,OTHER,,NETED,OUT,FID=<fid>,MSACCES=<password>.
```

2. Print one copy of wide output in duplex on 3-hole punched paper:

Send file OUTPUT immediately:

```
BEGIN,XEROX,,OUTPUT,xxxx.
```

Route file OUTPUT, including the dayfile, at end-of-job:

```
jobname,....
```

```
CHARGE,....
```

```
REWIND,OUTPUT.
```

```
LASER,OUTPUT,JDE=STDLND.
```

```
...
```

```
ROUTE,OUTPUT,DC=PA,TID=C,FC=XH,FID=xxxx,DEF.
```

```
...
```

3. Print 3 copies (the original and 2 extras) of file OUTFILE on unpunched paper:

```
BEGIN,XEROX,,OUTFILE,xxxx,COPIES=3,PAPER=PLAIN.
```

-or-

```
BEGIN,XEROX,,OUTFILE,xxxx,3,,PLAIN.
```

***** INTERFACING WITH OTHER COMPUTERS *****

*** Navy Laboratory Computer Network (NALCON) ***

The Navy Laboratory Computer Network (NALCON) is a subset of the larger network, MILNET, which enables computer users of one Navy Laboratory to access hardware and software resources at any other Navy Laboratory. Users can also interact with other computer systems linked to MILNET or ARPANET. Procedures exist among the Laboratories which simplify interactive access to these computer systems as well as file transfer among these computers.

For further information on NALCON and MILNET, contact User Services, Carderock.

*** NEMS ***

The NAVY Electronic Mail Service (NEMS) is an unclassified system which, for a prepaid annual subscription fee, will provide to the NAVY Department and their support personnel an electronic mail service. This service includes the following:

- . Defense Data Network (DDN) access and a mail box
- . Utilities for generating and managing electronic mail
- . File storage (50KB characters)
- . User and System support services

The NEMS facility provides the following communication facilities:

- . DDN access
- . Access to NEMS via GTE Telenet (Value Added Network) service

NEMS is currently based on Berkeley UNIX 4.1.

For more information about NEMS, contact User Services, Carderock.

*** PC Interface ***

```
**          XMODEM          **
**  Micro-to-Mainframe File Transfer  **
```

XMODEM is an implementation of the Christensen XMODEM protocol for our CDC computers. It provides a mechanism for transferring data files between two computer systems connected via asynchronous telecommunications lines. It increases the reliability of such transfers and provides a method for controlling the data flow between the connected computers.

With XMODEM, you can send files from your personal computer to the CDC for storage and later retrieval. You can also retrieve, for example, a FORTRAN source program from the CDC for editing at your PC, then send it back to CDC for compilation and execution.

To use:

```
ATTACH.XMODEM.
XMODEM,lfn,direction,file_mode.
```

where <lfn> is the local file name of the file to be transferred

<direction> is S (CDC will send to the PC)
R (CDC will receive from the PC)

<file_mode> is X (transfer "as is" - without translation)
D (translate ASCII to Display Code in receive mode -or-
translate Display Code to ASCII in send mode)
N (the data transferred is "NETED"-style ASCII)

For a more detailed explanation of XMODEM and its parameters as well as some examples, see "An Implementation of the XMODEM Protocol for DTNSRDC's Computer Systems", Robert W. Tinker, TM-18-84-09.

***** Appendix A *****

		*** DTNSRDC Character Set ***					note/name
display code	char- acter	punch 026	punch 029 if diff	7-track EXT BCD	9-track ASCII	9-track EBCDIC	
01	AAA	12-1		61	41	C1	
02	BBB	12-2		62	42	C2	
03	CCC	12-3		63	43	C3	
04	DDD	12-4		64	44	C4	
05	EEE	12-5		65	45	C5	
06	FFF	12-6		66	46	C6	
07	GGG	12-7		67	47	C7	
10	HHH	12-8		70	48	C8	
11	III	12-9		71	49	C9	
12	JJJ	11-1		41	4A	D1	
13	KKK	11-2		42	4B	D2	
14	LLL	11-3		43	4C	D3	
15	MMM	11-4		44	4D	D4	
16	NNN	11-5		45	4E	D5	
17	OOO	11-6		46	4F	D6	
20	PPP	11-7		47	50	D7	
21	QQQ	11-8		50	51	D8	
22	RRR	11-9		51	52	D9	
23	SSS	0-2		22	53	E2	
24	TTT	0-3		23	54	E3	
25	UUU	0-4		24	55	E4	
26	VVV	0-5		25	56	E5	
27	WWW	0-6		26	57	E6	
30	XXX	0-7		27	58	E7	
31	YYY	0-8		30	59	E8	
32	ZZZ	0-9		31	5A	E9	
33	000	0		12	30	F0	(sometimes 00)
34	111	1		01	31	F1	
35	222	2		02	32	F2	
36	333	3		03	33	F3	
37	444	4		04	34	F4	
40	555	5		05	35	F5	
41	666	6		06	36	F6	
42	777	7		07	37	F7	
43	888	8		10	38	F8	
44	999	9		11	39	F9	
45	+++	12	12-6-8	60	2B	4E	plus
46	---	11		40	2D	60	minus
47	***	11-4-8		54	2A	5C	asterisk
50	///	0-1		21	2F	61	slash
51	(((0-4-8	12-5-8	34	28	4D	left paren
52)))	12-4-8	11-5-8	74	29	5D	right paren
53	\$\$\$	11-3-8		53	24	5B	dollar
54	===	3-8	6-8	13	3D	7E	equal
55				20	20	40	blank
56	, , ,	0-3-8		33	2C	6B	comma
57	...	12-3-8		73	2E	4B	period

display code	char- acter	punch 026	punch 029 if diff	7-track EXT BCD	9-track ASCII EBCDIC		note/name
60	###	0-6-8	3-8	36	23	7B	pound
61	[[[7-8	12-2-8	17	5B	4A	l bracket (1)
62]]]	0-2-8	11-2-8	32	5D	5A	r bracket (1)
63	:::	2-8			25	6C	colon (2) (sometimes 16)
64	"""	4-8	7-8	14	22	7F	quote
65	---	0-5-8		35	5F	6D	underline
66	!!!	11-2-8	12-7-8	52	21	4F	exclam (1,3)
66	!!!	11-0		52	21	4F	exclam (1,3)
67	&&&	0-7-8	12	37	26	50	ampersand
70	' ' '	11-5-8	5-8	55	27	7D	apostrophe
71	???	11-6-8	0-7-8	56	3F	6F	question
72	<<<	12-2-8	12-4-8	72	3C	4C	less than (1,3)
72	<<<	12-0		72	3C	4C	less than (1,3)
73	>>>	11-7-8	0-6-8	57	3E	6E	greater than
74	@@@	5-8	4-8	15	40	7C	at
75	\\	12-5-8	0-2-8	75	5C	E0	reverse slant
76	^^^	12-6-8	11-7-8	76	5E	5F	circumflex
77	:::	12-7-8	11-6-8	77	3B	5E	semicolon (4)
55		6-8	0-4-8		20	40	blank (5)

Notes:

- (1) The six characters different between SCOPE 3.3 and NOS/BE are left bracket, right bracket, exclamation, apostrophe, less than, and at. The 026 punch is different for each. The 029 punch is the same except for alternate punches 11-0 and 12-0 (the display code associated with the 029 is different.) These 6 characters may have different graphics on a 200-UT-compatible terminal.
- (2) On 7-track tape, this becomes zero (display 33).
- (3) alternate punches.
- (4) Avoid a whole word of semicolon, which is a negative zero and is treated as an end-of-record.
- (5) On some terminals, this is transmitted as a binary zero. For these terminals, avoid putting this punch in columns 9-10, 19-20, ..., 79-80, as each will be interpreted as a zero-byte terminator.

*** ASCII Character Set ***

CHAR	ASCII (HEX)	EBCDIC (HEX)	DISPLAY (OCTAL)	CHAR	ASCII (HEX)	EBCDIC (HEX)	DISPLAY (OCTAL)
NUL	00	00		000	30	F0	33
SOH	01	01		111	31	F1	34
STX	02	02		222	32	F2	35
ETX	03	03		333	33	F3	36
EOT	04	37		444	34	F4	37
ENQ	05	2D		555	35	F5	40
ACK	06	2E		666	36	F6	41
BEL	07	2F		777	37	F7	42
BS	08	16		888	38	F8	43
HT	09	05		999	39	F9	44
LF	0A	25		:::	3A	7A	63
VT	0B	0B		:::	3B	5E	77
FF	0C	0C		<<<	3C	4C	72
CR	0D	0D		===	3D	7E	54
SO	0E	0E		>>>	3E	6E	73
SI	0F	0F		???	3F	6F	71
DLE	10	10		@@@	40	7C	74
DC1	11	11		AAA	41	C1	01
DC2	12	12		BBB	42	C2	02
DC3	13	13		CCC	43	C3	03
DC4	14	3C		DDD	44	C4	04
NAK	15	3D		EEE	45	C5	05
SYN	16	32		FFF	46	C6	06
ETB	17	26		GGG	47	C7	07
CAN	18	18		HHH	48	C8	10
EM	19	19		III	49	C9	11
SUB	1A	3F		JJJ	4A	D1	12
ESC	1B	27		KKK	4B	D2	13
FS	1C	1C		LLL	4C	D3	14
GS	1D	1D		MMM	4D	D4	15
RS	1E	1E		NNN	4E	D5	16
US	1F	1F		OOO	4F	D6	17
space	20	40	55	PPP	50	D7	20
!!!	21	4F	66	QQQ	51	D8	21
""""	22	7F	64	RRR	52	D9	22
####	23	7B	60	SSS	53	E2	23
\$\$\$	24	5B	53	TTT	54	E3	24
%%%	25	6C		UUU	55	E4	25
&&&	26	50	67	VVV	56	E5	26
'''	27	7D	70	WWW	57	E6	27
((28	4D	51	XXX	58	E7	30
)))	29	5D	52	YYY	59	E8	31
***	2A	5C	47	ZZZ	5A	E9	32
+++	2B	4E	45	[[[5B	4A	
,,,	2C	6B	56	\\	5C	E0	75
---	2D	60	46]]]	5D	5A	
...	2E	4B	57	^^^	5E	5F	76
///	2F	61	50	---	5F	6D	65

CHAR	ASCII (HEX)	EBCDIC (HEX)	DISPLAY (OCTAL)
grave	60	79	
aaa	61	81	
bbb	62	82	
ccc	63	83	
ddd	64	84	
eee	65	85	
fff	66	86	
ggg	67	87	
hhh	68	88	
iii	69	89	
jjj	6A	91	
kkk	6B	92	
lll	6C	93	
mmm	6D	94	
nnn	6E	95	
ooo	6F	96	
ppp	70	97	
qqq	71	98	
rrr	72	99	
sss	73	A2	
ttt	74	A3	
uuu	75	A4	
vvv	76	A5	
www	77	A6	
xxx	78	A7	
yyy	79	A8	
zzz	7A	A9	
{{{	7B	C0	61
	7C	6A	
}}}	7D	D0	62
~~~	7E	A1	
DEL	7F	07	

## ***** Appendix B *****

## *** Keypunch Characters ***

The standard keypunch character set is that of the IBM model 026 keypunch. However, there are ways of using a subset of the EBCDIC character set (see Appendix A for the characters allowed).

Source cards punched from the punch file on CDC computers will normally be 026 whether input was 026 or 029. The ROUTE command (page 5-12) is used to punch 029 mode at Central Site. There is a routine for conversion of an 026 deck to 029 for punching at some remote batch terminals (see CLIB/P: CV029).

Programs received on 7-track tape in EBCDIC are not handled by the automatic conversion, but must be converted by a special program.

## *** Use of Alternate Keypunch Characters ***

## ** For Jobs Submitted at Central Site or From a 1700 **

The 029 keypunch may be used to punch cards for the CDC, if the end-of-record card (or the job card) preceding those cards has 29 punched in columns 79 and 80. For example, if all cards in a job have been punched on an 029 keypunch, then 29 should be punched in the job card. However, if only the source cards have been punched with the EBCDIC characters, then a 29 should be punched in columns 79 and 80 of the end-of-record card (7/8/9) immediately preceding the source cards. The input for a succeeding record may be reset to 026 mode by punching 26 in columns 79 and 80 of the preceding 7/8/9 end-of-record. Note that all cards in a logical record must be of the same keypunch character set. (see Appendix A)

The most frequently used characters which differ between 026 and 029 keypunches are the following four characters ( ) = and +.

## ** For Jobs Submitted From a Medium-speed Terminal **

To be read by a medium-speed (200-UT-compatible) terminal, all cards in a job deck must be in the same punching, and the operator manually will select 026 or 029 mode. The 026 mode is standard. (Punch 29 in columns 79-80 of the job card to indicate the mode to the operator.)

## ***** Appendix C *****

## *** Printer Carriage Control ***

The first character of each line to be printed is called the carriage control character. It is not printed but, instead, is used to determine the vertical positioning of the printer before and after the rest of the line (characters 2 on) is printed. The channels refer to holes punched in a carriage control tape which is mounted inside the printer.

The following are the valid carriage control characters:

character	action before printing	action after printing
A	space 1	skip to top of next page
B	space 1	skip to last line of page
C	space 1	skip to channel 6
D	space 1	skip to channel 5
E	space 1	skip to channel 4
F	space 1	skip to channel 3
G	space 1	skip to channel 2
H	space 1	skip to channel 11
I	space 1	skip to channel 7
J	space 1	skip to channel 8
K	space 1	skip to channel 9
L	space 1	skip to channel 10
* 1	skip to top of next page	no space
2	skip to last line of page	no space
3	skip to channel 6	no space
4	skip to channel 5	no space
5	skip to channel 4	no space
6	skip to channel 3	no space
7	skip to channel 2	no space
8	skip to channel 11	no space
9	skip to channel 7	no space
X	skip to channel 8	no space
Y	skip to channel 9	no space
Z	skip to channel 10	no space
* + (plus)	no space	no space
* (blank)	space 1	no space
* 0 (zero)	space 2	no space
* - (minus)	space 3	no space

Note that top-of-page (A, 1) is a channel 1 punch; bottom-of-page is a channel 7 punch.

An invalid carriage control character is normally treated as a blank.

See page 22-2 for PM carriage control.

* - These are the normal carriage control characters used in FORTRAN and COBOL.

**** Perforation Skipping ****

The following carriage control characters control automatic skipping over the perforation between pages. Its success depends on how the paper is aligned in the printer. No printing takes place and columns 2 on are ignored.

character	action
Q	clear auto page eject (default)
R	select auto page eject

Q and R are ignored at some remote printers in which case the rest of the line will be printed.

**** Print Density Control ****

The following carriage control characters set the print density to 6 or 8 lines per vertical inch. No printing takes place and columns 2 on are ignored.

character	action
S	set 6 lines per inch (default)
T	set 8 lines per inch

S and T are ignored at remote printers where the print density is a hardware function in which case the rest of the line will be printed.

** User-programmable 580 Printer Control **

The CYBER 176 has model 580 user-programmable printers. Such a printer does not use a carriage control tape. It has a microprocessor plus memory. Format arrays are loaded into the printer memory by using the spacing code (SC) parameter on the ROUTE command for system-defined format arrays or by using the V carriage control.

A line with a V carriage control character has the following format:

col	contents
1	V
2	One of: 6 - 6 lines per inch (1 line) 8 - 8 lines per inch (1 line) C - 8 lines per inch (2 lines) Any other characters invalidate the array.
3 on	Defines a format array of up to 132 characters plus end-of-array terminator for 6 lines per inch or 176 characters plus end-of-array terminator for 8 lines per inch. The format array is made up of the following characters:

character	meaning
A	Top-of-form. (the array must begin with an A)
B	Channels 2 thru 11, respectively. Since
thru	other carriage control characters may
K	cause skips to other channels, each of
	these letters should appear at least
	once in the array.
L	Bottom-of-form.
O	End-of-array. Must be the last
	character but does not correspond to
	any line of the form.
blank	No channel. Blanks increase the number
	of lines in the form.

Any other characters invalidate the array.

Note that the V lines do not set the print density. Only the S and T lines do. On a non-580 printer, the V carriage control character causes a page eject with no line printed.

** Examples **

- 1) 6 lpi buffer with a 22-column array, implying a 21-line form.

column	123456789012345678901234
array	V6A B C D EFGHIJK O

- 2) 8 lpi buffer with a 12-column array, implying a 11-column form.

column	12345678901234
array	V8ABCDEFGHJKO

- 3) 8 lpi buffer with a 22-column array, implying a 21-line form.

column	12345678901234567
array	VCA B C D
	VC E F G H I JK O

## ***** Appendix D *****

## *** Internal Data Structure ***

1. Internal representation of Hollerith data is Display Code in the CDC CYBER but ASCII, EBCDIC or internal BCD in some other systems. Note that a Display Code zero is an octal 33 and not 00, which could cause trouble with masks.

Hollerith word	internal machine representation
<blanks>	55555555555555555555
<zeroes>	33333333333333333333
TAPE48	24012005374355555555

2. The character sequence for the CDC computers at DTNSRDC is the ASCII 63-character set (see Appendix A). Note that the alphabet precedes numbers and special characters for FORTRAN alphabetic compare (FTN5, Appendix A; FTN4, Appendix A). COBOL uses the ASCII6 collating sequence (numbers before letters) (COBOL, Appendix A). To use the COBOL6 collating sequence (letters before numbers), the U parameter must be included in the COBOL statement (page 14-12). The collating sequence for SORT/MERGE is specified by the user in the SORT/MERGE control statements (SORT5, 2-2 ff, 3-5, 5-4, Appendix A; SORT4, 4-11 thru 14, Appendix A).

3. The following table summarizes word lengths on various computers:

	CDC CYBER 170	B 7700	DEC VAX IBM 303x	CDC 200 CDC 180	UNIVAC 11xx IBM 7090
bits/word	60	48	32	64	36
digits/word	20 oct	12 hex	8 hex	16 hex	12 oct
char/word	10	6	4	8	6

This affects the conversion of programs in four areas:

- a. The degree of precision of operations is much higher. Therefore, convergence factors may be smaller in absolute value.
- b. Constants and data may be increased to greater accuracy by adding additional significant digits, up to 15 digits.
- c. Octal constants used in masking operations are generally affected and require alteration according to their intended use.

- d. Since words contain 10 characters per word instead of 6 or 4, DATA statements that store a string of Hollerith characters may position the characters in different relative positions in different words, except for the first 6 or 4 characters. All variable formats (whether read in as data or created by the programmer) should be checked.
4. The CDC CYBER uses some special bit configurations in floating point arithmetic to indicate indefinite and infinite operands. On the CYBER 750, an indefinite may be caused by 0/0 and an infinite by n/0 operation. On both CYBERs, either error could be caused by referencing program areas not initialized or areas overwritten due to inadequate storage reservation. The CPU will not do any further calculation if it encounters such a number and the job will abort with an error mode 2 or 4 (see page 15-4).
- + infinity 3777xxxxxxxxxxxxxxxx
  - infinity 4000xxxxxxxxxxxxxxxx
  - + indefinite 1777xxxxxxxxxxxxxxxx
  - indefinite 6000xxxxxxxxxxxxxxxx
- where 'x' is any octal digit, usually 0.
5. The word format of floating point numbers has 48 bits for the coefficient, 11 bits for the exponent plus 1 sign bit. The range of permissible numbers is  $10^{*-293}$  to  $10^{*322}$ . Integers may be 60 bits but must not exceed 48 bits ( $2^{*48} - 1$ , 15 digits) if they are to be used in multiplication and division operations. Integer multiplication uses a double precision hardware instruction. Integer division is performed by software using the floating point registers.

example	CDC CYBER	Burroughs 7700	IBM 303x
1	0000000000 0000000001	000000000001	00000001
-1	7777777777 7777777776	400000000001	FFFFFFFF
1.0	1720400000 0000000000	261000000000	41100000
		or 000000000001	
-1.0	6057377777 7777777777	661000000000	C1100000
		or 400000000001	
2.0	1721400000 0000000000	262000000000	41200000
		or 000000000002	
4.0	1722400000 0000000000	264000000000	41400000
		or 000000000004	

Note the difference in the format of negative numbers:

CDC CYBER	Burroughs 7700	IBM 303x
one's complement of absolute value	signed magnitude and exponent	two's complement of absolute value



6. Logical variables in the CYBER are expressed as -1 for TRUE and +0 for FALSE.
7. By default, the program field length in central memory is set to DEBUG (negative indefinite with addresses and some bits set for CYBER Interactive Debug). See page 8-6: PRESET/PRESETA.
8. Computer instructions are 15 or 30 bits long. Hence one CM word may contain up to four instructions. The No-Operation instruction (46000) is used to fill words when the next instruction cannot fit into the same word.

## ***** Appendix E *****

## *** Octal-decimal Conversion ***

Addresses and data occurring in memory dumps are octal.

octal	decimal
10	8
40	32
100	64
400	256
1000	512
2000	1024
4000	2048
10000	4096
20000	8192
30000	12288
40000	16384
50000	20480
60000	24576
70000	28672
100000	32768
140000	49152
200000	65536

Tables of octal-decimal and octal-hex are in SUG, Appendix B.

## *** CDC CYBER Real Floating ***

decimal	positive		negative	
.0001	1702643334	2726161031	6075134443	5051616746
.0005	1705406111	5645706520	6072371666	2132071257
.001	1706406111	5645706520	6071371666	2132071257
.005	1710507534	1217270244	6067270243	6560507533
.01	1711507534	1217270244	6066270243	6560507533
.05	1713631463	1463146315	6064146314	6314631462
.1	1714631463	1463146315	6063146314	6314631462
.5	1717400000	0000000000	6060377777	7777777777
0.0	0000000000	0000000000	7777777777	7777777777
1.	1720400000	0000000000	6057377777	7777777777
5.	1722500000	0000000000	6055277777	7777777777
10.	1723500000	0000000000	6054277777	7777777777
50.	1725620000	0000000000	6052157777	7777777777
100.	1726620000	0000000000	6051157777	7777777777
500.	1730764000	0000000000	6047013777	7777777777
1000.	1731764000	0000000000	6046013777	7777777777
5000.	1734470400	0000000000	6043307377	7777777777
10000.	1735470400	0000000000	6042307377	7777777777
1.0E15	2002706576	5114320000	5775071201	2663457777

*** Octal-decimal Fractions ***

octal	dec	octal	dec
.1	.125	.01	.015625
.2	.250	.02	.031250
.3	.375	.03	.046875
.4	.500	.04	.062500
.5	.625	.05	.078125
.6	.750	.06	.093750
.7	.875	.07	.109375

octal	dec	octal	dec
.001	.001953	.0001	.000244
.002	.003906	.0002	.000488
.003	.005859	.0003	.000732
.004	.007812	.0004	.000976
.005	.009765	.0005	.001220
.006	.011718	.0006	.001464
.007	.013671	.0007	.001708

octal	dec	octal	dec
.00001	.000030	.000001	.000003
.00002	.000061	.000002	.000007
.00003	.000091	.000003	.000011
.00004	.000122	.000004	.000015
.00005	.000152	.000005	.000019
.00006	.000183	.000006	.000022
.00007	.000213	.000007	.000026

** Conversion Examples **

octal number 1716 4135240123456773  
 decimal value  $2^{**}(-1) * .500$

.015625  
 .005859  
 .001220  
 .000061  
 .000015

-----  
 $2^{**}(-1) * .522780 = .26139$

octal number 2002 7065765114320000  
 decimal value  $2^{**}62B * .875$

.000  
 .011718  
 .001220  
 .000213  
 .000022

-----  
 $2^{**}62B * .888173 = 1.0E15$   
 $2^{**}50$

## ***** Appendix F *****

## *** Control Statement Field Lengths ***

DTNSRDC NOS/BE users should omit CM from their job card unless memory in excess of 47200 will be required. The system adjusts its FL automatically for the following products and system control statements:

ACCTRPT	44000	LABEL	2000
ADPCOST	33000	LIBRARY	1000
ALTER	5000	LISTMF	1000
ATTACH	5000	LOADPF	26000
AUDIT	5000		
		MAP	1000
BEGIN, REVERT	10000	MOUNT	2000
BKSP	1000	MSACCES	12000
		MSAUDIT	12000
CASE	10000	MSCHANG	12000
CATALOG	5000	MSFETCH	12000
CHARGE	3200	MSPASSW, MSPERMT	12000
CLEAR, RETAIN	400	MSPURGE	12000
CKP	13000	MSSTORE	12000
COMBINE	20000		
COMPARE	7000	NOTE	11000
COPY	20000		
COPYBF, COPYBR	20000	PAUSE	200
COPYCF, COPYCR	20000	PRNTSPY	40000
COPYCL	37000	PURGE	5000
COPYE	12000		
COPYF	12000	RECOVER	64500
COPYL, COPYLM	12000	RENAME	5000
COPYN	6000	REQUEST	2000
COPYR	12000	RESTART	25000
COPYRM	21000	RETURN	200
COPYSBF, COPYSF	3000	REWall	2000
COPYSR	3000	REWIND	200
		ROUTE	1000
DISPOSE	400		
DSMOUNT	2000	SISTAT	21000
DUMPF	26000	SKIPB, SKIPF	1000
		SYSBULL	2500
ESTMATE	37000		
EXTEND	5000	TRANSF	300
FILE	3000		
		UNLOAD	200
ITEMIZE	12000		
IXGEN	65000	VSN	1000
		WARNING	3000

*** Recommended Field Lengths ***

ABAQUS	162000 minimum	see page 20-1
APL	35000 minimum	see page 20-2
APT	65200 minimum	ATTACH,APT3.
	122000 minimum	ATTACH,APT4.
BASIC	40000	
COBOL	61000 minimum	
COBOL5	65000 minimum	
COMPASS	54000 minimum	
DDL	54000 minimum	
ECAP	70000	MSFETCH,ECAMP,UN=CSYS.
EDITLIB	45000 minimum	
FORM	71000 minimum	
FTN4	47200 minimum	(prefer 54000)
FTN4,D	61000 minimum	
FTN5	61000 minimum	
F45	75000 minimum	
GPSS v1.3	70000 minimum	MSFETCH,GPSS,UN=CSYS.
v1.2	130000 minimum	MSFETCH,GPSS,GPSS12,ID=CSYS.
LCS	77000	see page 14-8
LOADER, LOAD	30000 minimum	(will get more as needed)
MNF	53000	see page 13-37
NETED	20000	ATTACH,NETED.
OMNITAB	170000	see page 20-12
PASCAL	45000 minimum	
PLI	64000 minimum	
QU (Query Update)	67000 minimum	
RATFOR	45000	ATTACH,RATFOR.
RUN	45000	ATTACH,RUN2P3.
SIMII5 (Simsript II.5)	75000 minimum	see page 20-14
SNOBOL	76000 minimum	ATTACH,SNOBOL.
SORT/MERGE	60000 minimum	
S2K30, S2K280	53000 minimum	see page 20-25
UPDATE	40000	

Note: some products may require either CM or both CM and RFL.

For cataloged procedures, see CLIB/P.

For routines on UTILITY library, see page 10-25, and CLIB/U.

For special products, see Chapter 11.

## ***** Appendix G *****

## *** Glossary ***

**Access number**

Each job order number is assigned an access number which is used on the batch CHARGE card (see 5-5) and Intercom LOGIN (see 4-2).

**Alphameric** A letter (A-Z) or a digit (0-9). Also called alphanumeric.

**Catalogued procedure**

A previously-defined sequence of control statements for performing a task. A catalogued procedure may be executed using the BEGIN control statement.

**CLIB, CLIB/N, CLIB/P, CLIB/U**

Used throughout this manual to refer to "Computer Center CDC Libraries", CMLD-84-11 CLIB/NSRDC, CMLD-84-12, CLIB/PROCFIL, CMLD-84-13, CLIB/UTILITY, CMLD-84-14, respectively.

**CCRM**

Used throughout this manual to refer to the "Computer Center CDC Reference Manual", CMLD-84-10 (this manual).

**CDC CYBER**

Used to refer to any of the CDC CYBER 170 computers at DTNSRDC: CYBER 176, CYBER 170 model 750. These computers are also referred to by their mainframe letters: MFE (CYBER 750), MFF (CYBER 176).

**Control card record****Control statement record**

The first group of statements/cards in a batch job, ending with a card having 7/8/9 multi-punched in column 1 (if a card deck) or an end-of-record (if created interactively). These are all the control statements to be processed during the job. Any additional records, such as a source program or data, follow the control statement record.

**Dayfile, batch**

As a batch job is being run, a permanent record of the job activity is created. This is called the dayfile, a copy of which is printed at the end of each batch job. The dayfile includes a list of all control statements executed, any system- or program-generated messages and a summary of the system usage including the estimated basic charge. This charge does not include card reading/punching or line printing. (See POLICY for the current rates.) Each message has the time-of-day it was written.

**Dayfile, Intercom**

As commands are executed during an Intercom session, messages are generated similar to those in batch. They are collected and printed at the terminal, usually at the end of each command, though some may be printed during the execution of a command. Except for logout, the dayfile messages are not time-stamped.

**<eor>, <eoi>**

Used in examples in this manual to represent an EOR (end-of-record) or EOI (end-of-information). In a card deck, <eor> is a multi-punched 7/8/9 in column 1; <eoi> is a multi-punched 6/7/8/9 in column 1.

**Field length (FL)**

The amount of memory occupied by a program. Addresses in a program are relative to the start of the field length, called the reference address (RA). A program occupies from RA+0 thru RA+FL-1. Thus, a user never needs to know the actual location of the program in memory.

**Logical record**

A group of information separated from other information in a file by end-of-record's. In a card deck, the 7/8/9 card is the end-of-record.

**Login name**

A 4-to 10-character name assigned by Code 189.3 or 1892.1. The first four characters are your user initials, the remaining are normally the first six letters of your last name. It is used to identify you when you LOGIN to Intercom.

**POLICY**

Used throughout this manual to refer to the "Computer Center Policy" manual.

**User initials**

A 4-character ID assigned to each user by Code 189.3. This is used to identify jobs, for charge authorization, to identify permanent files and magnetic tapes, etc.

## ***** Appendix H *****

## *** Computer Reference Centers ***

A complete library of all CDC and many other computer product documents (Appendix I) may be consulted in Code 1892.1 (Bldg. 17, Room 100). Most may also be consulted in Code 1892.2 (Annapolis, Bldg. 100, Room 1-KK). Subsets of frequently used manuals are available for reference at a number of other sites. These Computer Reference Centers are listed below:

code	name	location
1524	A. Reed	Bldg 3, Rm 214
1544	E. Caster	Bldg 16, Rm 105
1576	W. Meyer	Bldg 18, Rm 201E
1605	E. Rogers	Bldg 7, Rm 118
175	P. Roth	Bldg 19, Rm B162
1806	C. Lubin	T-32
182	A. Camara	Bldg 191, Rm 121
1843	R. van Eseltine	Bldg 192, Rm 128
1844	S. Wybraniec	Bldg 192, Rm 121
1856	R. Milam	Bldg 17, Rm 230
187	M. Christie	Bldg 17, Rm 214
1892.1	J. Strickland	Bldg 17, Rm 100
1892.2	D. Sommer	Bldg 100, Rm 1-T, Annapolis
1892.3	L. Minor	Bldg 17, Rm 105B
1926	K. Jones	Bldg 15, Rm 112
1965	J. Niemiec	Bldg 3, Rm 338
522	DTNSRDC library	Bldg 1, Rm 211
6080	B. Pierce	Bldg 121, Rm 132
UERD	J. Krezel	Portsmouth, Va.
03F2	J. Given	NAVSEA - Bldg NC2
5033	R. Saenger	NAVAIR - Bldg JP2
5707	H. Bryant	NRL, Bldg 35
	G. Vega	Bay St. Louis, Miss.
	S. Stefoncik	NWSC, Crane, Ind.
TEC	J. Tom	Hud, Room 8226



## ***** Appendix I *****

## *** Computer Reference Manuals ***

name	publication number
Computer Center Policy	.
Computer Center Introductory Reference Manual for CDC CYBER	CMLD-84-09 Rev0
Computer Center CDC Reference Manual	CMLD-84-10 Rev0
Computer Center CDC Libraries	CMLD-84-11
Computer Center CDC Libraries/NSRDC (Subprograms)	CMLD-84-12
Computer Center CDC Libraries/PROCFIL (Procedures)	CMLD-84-13
Computer Center CDC Libraries/UTILITY (Programs)	CMLD-84-14
Computer Center Mass Storage System User's Guide	CMLD-82-19
An Implementation of the XMODEM Protocol for DTNSRDC's CDC Computer Systems	TM-18-84-09
ADP Glossary	NAVSO P-3097
Computer Program Documentation Standards	SECNAVINST 5233.1B of 25 Jan 1979
Programming Calcomp Pen Plotters	1006D 5K 1277
Calcomp Functional Package Business	1011A 6C 1069
Calcomp Functional Package Crvt	1014 6C 0769
Calcomp Functional Package Drafting	1012A 3C 0670
Calcomp Functional Package General	1013A 3C 0670
Calcomp Functional Package Scientific	1015B 6C 170
Calcomp Three-D Manual	1002B 5M 769
DISSPLA Pocket Manual	
DISSPLA User's Manual	
Tektronix Plot 10 Terminal Control System User's Guide	062-1471-00
Tektronix Plot 10 Advanced Graphing II User's Guide	062-1530-00
Plot-10 Easy Graphing for 4027 Color Graphics Terminal User's Manual	.
Plot-10 Easy Graphing for 4027 Color Graphics Terminal User's Reference Card	.
System 2000 manuals	
LSM Define Language	1010
LSM Administrative Facilities	1013
LSM COBOL PLEX	1020
LSM FORTRAN PLEX	1022
LSM Quest Language	221126
Report Writer	221171
Messages and Codes	1090
Newsletter for CDC R3.0	221175

CDC manual	publ. nos.
APL Version 2 Reference Manual	60454000 A
APT Reference Manual Version 2 (APT III)	60174500 C
Automatic Programmed Tooling System (APT IV) Version 2 Reference Manual	17326900 A
Basic Version 3 Reference Manual	19983900 G
COBOL Version 4 Reference Manual	60496800 D
COBOL Version 5 Reference Manual	60497100 J
COBOL Version 5 User's Guide	60497200 D
COBOL 4/5 Conversion Aids	19265021 B
COMPASS Version 3 Reference Manual	60492600 G
CYBER Interactive Debug Version 1 Reference Manual	60481400 C
FORM Reference Manual	60496200 C
FORTRAN Common Library Mathematical Routines Reference Manual	60498200 C
FORTRAN Extended Version 4 Reference Manual	60497800 G
FORTRAN Version 5 Reference Manual	60481300 E
FORTRAN Version 5 Common Library Mathematical Routines Reference Manual	60483100 B
FORTRAN 4 Extended to Version 5 Conversion Aids	60483000 B
FTN Extended Debug User's Guide	60498000 A
GPSS	84003900 C
Intercom Version 4 Reference Manual	60494600 H
Intercom Interactive Guide/COBOL Users	60495100 B
Intercom Interactive Guide/FTN Users	60495000 B
Loader Version 1 Reference Manual	60429800 G
NOS/BE 1 Reference Manual	60493800 M
Record Manager Basic Access Methods 1 (COMPASS)	60495700 F
Record Manager BAM User's Guide	60495800 C
Record Manager COBOL 4 User's Guide	60496000 A
Record Manager FTN4 User's Guide	60495900 A
Reference Manual	60100000 AH
SCOPE Version 3.3/3.4 Conversion Aids	60358200 F
SCOPE 3.4 User's Guide	60358700 A
Sort/Merge Versions 4 (and 1) Reference Manual	60497500 F
Sort/Merge Version 5 Reference Manual	60484800 A
UPDATE Reference Manual	60449900 D

## ***** INDEX *****

Note - NOS/BE system control statements are flagged with ".  
Intercom commands are flagged with @.  
UPDATE directives begin with *.  
Compiler options are flagged with \$.

Primary references are flagged with an asterisk after the  
page number, for example, 1-1*.

## A

\$A	13-30
AAMLIB	18-1*
ABAQUS	20-1*
Abbreviation	i
Abort	5-6, 13-4, 13-28, 13-30, 15-7
Absolute	4-22
Absolute file	14-11
Absolute module	8-19
Absolute overlay	17-9
Access level	17-3*
Access number	5-5, 7-1, 7-2, g-1*
Account (AC)	9-5, 9-7
Accounting	5-5
"ACCTRPT	F-1
Acoustic coupler	4-2*
ADD (EDITLIB)	17-3*
ADDEXT	9-13
*ADDFILE	16-3*
Address (memory)	g-2*
ADP Control Center	2-1, 2-2
"ADPCOST	F-1
ADVANCING (COBOL)	14-7, 14-18
Aerospace Research Laboratories (ARL)	18-4*
ALGOL5	20-2*
Alphameric	9-7, g-1*
Alphanumeric	g-1*
"ALTER	9-6, 9-12, F-1
Alter	9-5
Anlog	20-9
Analysis of variance	19-4, 20-23
Analysis, Structural	20-10
\$ANSI	13-31

## A (continued)

ANSI	13-2, 13-38, 13-39, 14-12, 21-1
ANSI COBOL	14-2, 14-1, 14-11
APL	20-2*
"APT	F-2
APT3, APT4	20-3*
Argonne	18-4*, 18-5, 19-7
ARLNALG	18-4*
ARPANET	24-1
Array	13-11
ARRIBA	19-4*
ASCII	3-3, 4-27*, 11-2, 11-6, 24-3
ASCII terminal	1-3
ASCII 63-character set	D-1
ASCII6	14-18, D-1
Assembler	20-7
@ASSETS	4-6*, 4-19*
ASSIGN (FTN5)	13-15
Asynchronous	24-3
AT END (COBOL)	14-14
"ATTACH	4-18, 9-6, 9-10, 9-14, F-1
Attach	9-3, 13-23
Attributes	10-3
"AUDIT	9-5, 9-15, F-1
Audit	6-3
	10-3, 11-5
Auto page eject	C-1*
Automatic purge	9-3, 9-4
Automatically Programmed Tools (APT)	20-3*

## B

\$B	13-2, 13-30, 14-3, 14-11
Backspace	12-5
Backspace one character	4-5*
Backup	4-28, 9-15
Back-up	11-1
BAMLIB	18-1*
BANDIT	20-11
Banner	5-6
Banner card	2-3
"BASIC	20-7*, F-2
Basic	4-9, 19-7, 22-6
BASLIB	18-1*

## B (continued)

@BATCH	4-11*, 4-28
Batch	3-2, 5-9
Batch dayfile	g-1*
Batch job	1-5, g-1*
Batch, remote	3-3*, 4-6
Batch, remote commands	3-4
BCD	1-3, 2-2, 3-3, 11-6, 21-1, D-1
BCD to EBCDIC	21-1*
"BEGIN	6-1*, 6-2, 9-5, F-1, g-1
Begin	19-8
BEGIN, SEND	4-17
BENDIX (APT3)	20-3
Bessel functions	18-5, 18-7
BFS, buffer length	5-11
BGNPL	18-18
BIMED	19-1
BIMEDP	19-5*
Binary	4-8, 11-1, 11-3, 11-5, 13-28
Binary deck	13-2
Binary file	12-1
Binary programs	1-1
Binary tape	23-2
BIT8LIB	18-1*
"BKSP	12-5*, F-1
SBL	13-2, 13-30
Blank common	13-29, 13-33
BLOCK CONTAINS	14-14, 14-17
BLOCK DATA	8-7, 8-11, 13-27
Block time	3-8
Blocking	11-5*
Box-Jenkins	19-4*, 20-23
BT, block type	5-11
Buffer	9-2, 13-7, 13-21, 13-28, 13-35, 15-5, 18-12
BUFFER IN	13-7, 13-16, 13-29, 18-9
BUFFER OUT	11-3, 13-7, 13-16
Bulletin	4-2
Bulletin, system	2-1, 4-4*, 5-9
Burroughs	11-6
Burstable list	13-2, 13-30
BYE (EDITOR)	4-9

## C

CALCFN	18-13*, 22-4
CALCIBL	9-16
Calcomp	18-15, 18-16, 18-17, 22-4
Calcomp plotters	18-10
Calcomp 936	23-3
Calculator mode	4-26*
CALC3D	18-16*
CALC936	18-12*
*CALL	16-3*
Calls	8-2
CALL/CANCEL	14-5
Cancel	4-5
CARDS	11-6
CARDS, CARDS2	2-2
Carriage control	3-3, 14-15, 21-2, C-1*
Carriage return (CR)	4-5, 4-29
"CASE	F-1
"CATALOG	4-18, 9-6, 9-10, 9-14, F-1
Catalog	9-3, 11-5, 13-23
Catalogued procedure	6-1*, g-1*
CCL	4-26, 9-3, 21-5
CCRM	g-1*
CDC CYBER	g-1*
CDC 200-UT	3-3
CDC 734	3-3
Center Notes	2-1
Central memory (CM)	1-1, 5-3*
Central processing unit (CPU)	1-1*
Central Site	2-2
Certify, tape	11-8*
Change field length	4-6, 4-19
Change time limit	4-6, 4-19
Channels	1-1*
CHAR	13-16
Character manipulation	18-7, 18-8
Character set	A-1
CHARACTER (FTN5)	13-15, 13-16
Characteristics, tape	11-1
"CHARGE	5-5*, F-1
Charge	2-2, 3-2, 5-1, 5-6, 9-4, 10-1
Checkpoint	20-4*, 20-6
CHEKPTX	20-4
Christensen XMODEM Protocol	24-3
CID (Cyber Interactive Debug)	13-31
Circuit Analysis	19-4

## C (continued)

CIVCO	19-4*
"CKP	20-4, F-1
Classified	1-1, 3-2, 5-4, 5-6, 7-1
Classified job card	7-5*
Classified jobs	7-4*
Classified microfiche	7-3*
Classified permanent files	7-4*
Cleaning, tape	11-8*
"CLEAR	5-7*, F-1
CLIB	g-1*
Close files	13-22
CLOSMS	13-28
CM (central memory)	5-3*, 14-4
CM, conversion mode	5-11
"COBOL	F-2
COBOL	9-16, 14-1, 18-1, 20-15, 20-18, 20-25, 20-30
COBOL calling FORTRAN	14-10
COBOL 4	14-11*
COBOL 5 versus COBOL 4	14-5
COBOL4 to COBOL5 (LCS)	14-8*, 21-1*
"COBOL5	F-2
COBOL5	14-2*, 14-1, 18-1
COBOL6 (collating sequence)	D-1
Coded file	11-1, 11-3, 11-5, 12-1
Collating sequence	13-16, 14-5, 14-12, 14-18
\$Collating sequence (CS)	13-2
Color terminal	22-7, 22-8
Color, cards	5-1
COLSEQ	13-2
COLTMC (APT3)	20-3
COM (computer output microfiche)	5-15, 11-2, 23-1
"COMBINE	12-5*, F-1
*COMDECK	16-3*
Command time limit	4-19
"COMMENT	5-7*, 21-4
Comment	5-1, 13-28
Common	8-10
Communications with operator	21-2*
COMPAR	12-10*
"COMPARE	12-5*, F-1
Compare error	2-2
Compare files	12-10
"COMPASS	20-7*, F-2
Compatibility	14-17

## C (continued)

Compilation errors	14-10
*COMPILE	16-3*
Compile	16-1
Compiler directive	13-14
Complaints	2-1
Complement	D-2
Complementary dayfile	5-14*
Complex	13-28
Compressed	16-1
COMPRS	18-17
Computational	14-16
Computer Center Notes	2-1
Computer Users Forum	2-1
COMP-1	14-10
COMQ	23-1*
Condense	17-9*
Confidential	5-6
CONMIN	18-4*
CONNEC	4-8
@CONNECT	4-6, 4-8
Constrained optimization	18-4
Consultation	2-1*
CONTENT (EDITLIB)	17-4*
Contents, binary	12-4, 17-3
@CONTIN (remote batch)	3-4
Continuation line	13-11
Continued command	9-3
Continued directives	8-12
Continuous system simulator	20-9
Contouring	22-4
Control statement	5-1, 5-2*, 6-1, F-1
Control statement parameters (COBOL5)	14-3*
Control statement parameters (COBOL)	14-11*
Control statement parameters (FTN4)	13-30*
Control statement parameters (FTN5)	13-2*
Control statement record	g-1*
Control statements	3-2, 4-11
Control (CN)	9-7
Conversion	14-13, 21-1, B-1
Conversion, octal-decimal	E-1
Convert	2-2, 12-13, 19-3
"COPY	12-5*, F-1
*COPY	16-3*
Copy	19-3
"COPYBF	12-1*, F-1



## C (continued)

COPYBFR	12-1, 19-3*
COPYBLK	9-1, 12-8
"COPYBR	12-1*, F-1
"COPYCF	12-1*, F-1
"COPYCL	F-1
"COPYCR	9-10, 12-1*, F-1
"COPYE	12-6*, 12-13, F-1
"COPYF	12-7*, 12-11, F-1
"COPYL	12-2*, 12-12, 13-29, F-1
"COPYLM	12-2*, F-1
"COPYN	12-3*, 12-12, 13-29, F-1
"COPYR	9-11, 12-7*, 12-11, F-1
"COPYRM	12-8*, 12-13, 12-14, F-1
COPYRM	9-1, 12-1
COPYS	12-1
"COPYSBF	12-9*, F-1
"COPYSF	12-9*, 12-11, F-1
"COPYSR	12-9*, F-1
Core	14-4
Core image	4-22, 8-1
Core-to-core	13-13
CP time limit	15-7
CPU	14-4
CPU time	4-6
Creation	16-1
Creation date, tape	11-11*
Credit	19-11
CRM (Cyber Record Manager)	13-33
Cross reference	13-36, 15-5
\$CS (collating sequence)	13-2
CSOWN (FTN5)	13-2, 13-18
CTRL H	4-5
CTRL S	4-5
CTRL X	4-5
Current users (Intercom)	4-20
Curve fit	18-7
CVT360	19-3*, 21-1
CV029	2-2, 4-28, 5-16, 21-1, B-1
CYBER	1-1*, g-1*
CYBER Interactive Debug (CID)	13-31, D-3
CYBER 176	1-1, g-1
CYBER 750	1-1, g-1
Cycle (CY)	9-7

## D

\$D	13-30, 14-11
Data	13-11
Data base	20-8
Data Base Management System (DBMS)	20-13, 20-25, 20-30
Data interchange	11-3*
Data library, UPDATE	16-1
Data management	20-8
Data set	4-2*
Data tone	4-2*
Data 100	3-3
Datagraphics Autocom Recorder	11-2
Dawson's Integral	18-5
"DAY	4-19*
Dayfile	3-2, 4-17, 4-19, 5-6, 5-14, 13-21, 14-11, 15-2, 15-5,
	15-7, 21-4, 23-5
Dayfile message	13-28
Dayfile (Intercom)	g-2*
Dayfile, batch	g-1*
Dayfile, complementary	5-14*
\$DB	13-3, 13-16, 13-31, 14-11, 14-19
DBUGLIB	18-1*
DC (ROUTE)	5-13*
"DDL (Data Description Language)	20-8*, F-2
Debug	8-6, 13-3, 13-14, 13-31, 13-35, 15-5
DEBUG (COBOL)	14-16
Debug (D)	13-30
Debug, Cyber Interactive (CID)	13-30
Debug, Interactive (ID)	13-3
DEC tape	11-6
DEC VAX	3-3
*DECK	16-3*
Deck	2-2, 4-28
Deck, binary	13-2
Decode	13-35
Default length	13-35
Deferred routing (DEF)	5-12
@DEFINE (remote batch)	3-4
*DELETE	16-3*
DELETE (EDITLIB)	17-3*
DELETE (EDITOR)	4-5, 4-9
DELETE (NETED)	4-8
Density	11-1, 11-2, 11-6

## D (continued)

Density, print	13-5
Density, tape	11-11
Dependency	5-4, 5-10*
Device set	5-3, 7-1, 9-13, 9-13, 10-1
Diablo 630	4-27*
Diagnostics	13-21, 13-31, 13-36, 13-38, 14-10, 14-12
Diebold tape	11-7*, 11-9
Differences (FTN4 vs FTN5)	13-11
Differential equations	18-5, 18-8, 20-9
Digital	20-9
Digitizer	22-4*
Dimension	13-28
Directives (compiler)	13-14
Directives (EDITLIB)	17-1*
Directives (SEGLOAD)	8-10*
Directives (UPDATE)	16-3*
Dirt, tape	11-8*
@DISCARD	4-18*
Disconnect	4-5
@DISCONT	4-8
Disk	13-7
Disk pack	9-2, 9-13
"DISPLAY	4-26*
Display	21-4
Display code	5-1, 24-3, A-1, D-1
Display (PAGE)	4-24
"DISPOSE	F-1
Disposition	5-13, 21-2
DISSPLA	18-10, 18-17, 18-18, 18-19, 22-3
DISSPOP	18-19
@DIVERT (remote batch)	3-5
Division by zero	13-27
"DMP	4-6, 5-7*
DMP	13-26
DMPX	15-2, 15-5
DMSLIB	18-1*
DMS170	20-8*
\$DO	13-3
DOCGET	6-4, 19-8
Documentation	19-8
DONEPL	18-18
DOSSIER	5-8, 13-23
Double buffering	13-35

## D (continued)

Double precision	13-27, 13-28, 19-3, 21-1
Double precision to single precision	21-1*
DO-loop	13-3, 13-13
Drafting, Calcomp	18-13
@DROP	3-5, 4-12*
SDS	13-3
"DSMOUNT	9-14, F-1
DUM	9-13
Dummy arguments	13-16, 13-27
Dump	5-7, 9-15, 15-2
Dump (Fortran)	15-5
"DUMPF	9-13, 9-15, F-1
Dump, file	15-8
Dump, Post Mortem	13-3, 13-9*, 13-16
Dynamic	13-33
Dynamic load	8-2
Dynamic storage allocation	13-29

## E

\$e	13-3, 14-11
EAM	2-2
Easy Graphing	22-7, 22-8
EBCDIC	2-2, 11-2, 11-6, 11-11, 11-13, 12-13, 21-1, B-1
EBCDIC to BCD	21-1*
ECAP	19-4*, F-2
EDIT (NETED)	4-8
Edition, tape	11-11*
"EDITLIB	8-4, 15-8, F-2
EDITLIB	8-3, 9-3, 14-11, 17-1*, 17-5, 17-9
@EDITOR	4-7, 4-9*, 4-11
Editor	4-5, 4-26, 4-29, 21-5
EDITOR, re-enter	4-9
@EFL	4-19*
Eigensystem	18-7
Eigenvalue	18-4, 20-1
EISPACK	18-4*
\$EL	13-3, 13-31
ELBOW	19-4*
Elliptic integrals	18-5, 18-7
Emulator	3-3
ENCODE	13-35
End around search	12-3

## E (continued)

END statement	13-12
@END (remote batch)	3-4, 3-5
ENDPL	18-18
ENDRUN (EDITLIB)	17-4*
End-of-file	4-5, 13-13, 14-18, 21-5
End-of-information	4-5, 5-1*
End-of-job	5-7
End-of-record	4-5, 5-1*, 21-1, 21-5*, B-1, g-2*
ENTER	14-10
ENTRY	13-14, 13-28
EOF	4-5, 13-17, 21-5
<eoi>	g-2*
EOR	4-5, 21-5
<eor>	g-2*
Equated files	13-7, 13-8*
SER	13-31
Erase	11-3
ERR (tape)	15-2
Error list	13-3
Error message	13-29, 13-36
Error message (Fortran)	13-21
Error message (medium-speed)	3-7
Error mode	D-2
Error termination	13-3
Error traceback	13-33
@ERRORS	4-19*
Errors (COBOL)	14-12
Errors (Fortran)	13-22, 15-5
Errors (loader)	4-5, 15-2
Errors (mode)	15-4, 15-5
Errors (MSS)	15-3
Errors (NOS/BE)	15-5
Errors (permanent file)	9-9
Errors, mode	5-8
ESC key	4-5
Estimate	11-4
"ESTMATE	9-16, F-1
SET (error termination)	13-4
@ETL	4-19*
Even parity	12-1
@EVICT	3-6, 4-12*
Exact copy	12-6
Examine	14-7
Execute	4-21, 8-4

## E (continued)

Execution	5-4
Execution error	14-19
Execution memory	14-4
Execution time	14-4
"EXIT	5-7*, 15-7
"EXIT(S)	5-7*, 6-1, 13-30
"EXIT(U)	5-7*
Expiration date	9-4
Exponent	D-2
Exponential integral	18-5
Exponentiation	13-12
"EXTEND	9-6, 9-11, F-1
Extend (EX)	9-7, 11-5
External	8-4
EZGR	22-8

## F

False	D-3
Fatal error	13-21, 15-2
FC (ROUTE)	5-14*
FDUMP	15-8
@FETCH	4-18*
FID (ROUTE)	5-13*
Field length (FL)	3-2*, 4-6, 4-19, 5-7*, 8-9, D-2, F-1, F-2, g-2*
FIELDATA	14-5
"FILE	5-11*, F-1
File	9-1, 9-16, 11-3, 20-15, 20-19
File description (COBOL)	14-14
File dump	15-8
File identification	5-13
File Information Table (fit)	5-11
File manipulation	5-7, 8-4, 12-1, 17-3, 18-7
File name substitution	14-16
File Organization (fo)	5-11, 9-7
File replacement at execution	13-8*
File size	9-4
File transfer, micro-to-mainframe	24-3*
@FILES	4-6, 4-11, 4-19*
Files	10-1
Files, save	4-18
Files, XEQ	4-21
Film	23-1
FILMPL	5-13

## F (continued)

@FIND	4-13, 4-14*
FINISH (EDITLIB)	17-3*
Finite element	20-1, 20-10
Flag	8-9
Flaw	9-13
Floating point	D-2, E-1
Flush, buffer	13-28
FMILL (APT3)	20-3
"FORM	F-2
Form	9-1, 9-16, 9-17, 12-1
Format	13-11, 13-13
Format (EDITOR)	4-9
FORMAT (Fortran)	13-16
format, tape	11-6
forms	5-15, 21-2
forms code (route)	5-12*, 5-15
forms code (ww)	23-1
forms overlay	23-4
Fortran	4-7, 4-8, 4-26, 8-4, 11-3, 13-1, 13-37, 18-1*, 20-15,  20-18, 20-25, 20-30, 21-1, D-1
Fortran errors	15-5
Fortran 77	13-38, 13-39
FORTTRAN-X	14-10
FOUO	5-6
Fourier Transform	18-7
Fractions	E-2
FTN4 to FTN5	21-1*
FTN4/FTN5, intermixed	13-16
"FTN4, FTN	13-30*, F-2
"FTN5	13-1, 13-2*, F-2
FTN5LIB	18-1*
Funds	5-5
FUNPACK	18-5*
FZ (-FZ)	14-6
"F45 (FTN4-to-FTN5)	13-19, F-2

## G

Gamma function	18-7
GE (6250)	5-3, 11-11, 11-13
GECENT (APT3)	20-3
GETPARM	13-10*, 13-22

## G (continued)

GIVING (COBOL)	14-15
Global	8-8, 8-10
\$GO	13-4, 13-31
GO TO-less	13-39
@GO (remote batch)	3-4, 3-5
GPSS	20-9*, F-2
Grandfather-father-son	11-1
Graphic	22-6, 22-7, 22-8
Graphic input (GID)	22-4
Graphics, interactive	22-1, 22-2
Gripe	2-1
	6-8

## H

@H (remote batch)	3-5
Hardware	1-2, 1-3, 1-4, 5-4
Harris 1600	3-3
Header, tape	11-14
Heat transfer	20-1
HELP (PAGE)	4-23
High-speed terminal	21-2
Hollerith	13-11, D-1

## I

\$I	13-4, 13-31, 14-3, 14-11
IBM	11-6, 11-14, 14-13
ICHAR	13-16
ID	9-5
ID (MSS)	10-5
IDDS	22-9*
Ideal post processor	22-4
*IDENT	16-3*
Identifier	16-1
IF	13-12
IF-THEN-ELSE	13-15, 13-38
Illegal instruction	15-4
Image post processor	22-4
IMPLICIT (FTN5)	13-18
IMSL	18-5*
INCLUDE	8-10, 8-11
Indefinite	13-27, 13-28, 15-4, D-2
INDEX (FTN5)	13-18



## I (continued)

Indexed sequential	9-1*, 9-16, 14-16
Infinite	13-27, 13-28, 15-4
Infinite retention	9-4
Informative	13-21
Inhibit noise brackets	11-3, 11-12
Initialize	9-13, 14-6, 14-19
Initials, registered	5-3
Input	13-35
Input file	14-14
Input queue	3-2*, 5-4, 5-10, 5-13, 5-13
*INSERT	16-3*
Inspect	14-7
Instruction	D-2
Integer	D-2
Integer programming	19-4
Integration	18-7
Interactive	1-5, 4-1, 22-2
Interactive Data Display System (IDDS)	22-9*
\$Interactive Debug (ID)	13-3
Interactive graphics	22-1, 22-9
Interactive Sort/Merge	20-19
Intercom	1-5, 3-3, 4-1*, 4-9, 7-2, 9-5, 9-9, 13-35
Intermixed FTN4/FTN5	13-16
Internal BCD	D-1
Interpolation	18-5
Interpreter	2-2
Interrupt key	4-5
INTRINSIC	13-14, 13-15
Invalid	9-4, 13-28
IO time limit	15-7
"ITEMIZE	12-4*, 15-8, F-1
"IXGEN	F-1
I/O	13-13, 13-29, 18-9

## J

@J (FIND)	4-11, 4-13, 4-14*
Jacobian elliptic functions	18-7
JANUS	1-5*
Job	1-5, 3-2, 4-13, 4-14, 4-15, 5-3, 5-4, B-1
Job card	5-3*, 5-4, 11-1
Job card, classified	7-5*

## J (continued)

Job Descriptor Table (JDT)	3-2*
Job order number	g-1
Job processing	3-2*, 5-1
Job rerun	15-6*, 15-7
Jobs, classified	7-4*

## K

Keypunch	B-1
Keywords	5-2*, 9-7
@KILL	4-12*

## L

\$L	13-4, 13-31, 14-3, 14-12
"LABEL	11-11*, F-1
Label	11-1, 11-3
Label records (COBOL)	14-14
Labelled	11-2, 11-14
"LASER	23-5*
"LCS (COBOL 4 to COBOL 5)	14-8*, F-2
LDSET	4-21, 5-11, 8-6*, 11-3, 18-3, 19-1, D-2
Left display code	8-13
LEGVAR	13-27
Length	13-27, 13-35
Level 414-380	19-11
Level 508, 461, 439, 434, 420, 414-380	19-10
Level 552	19-9
LF	13-23
"LGO	8-4*
LGO	4-21, 13-2, 14-4
LIB	8-6
LIBLOAD	4-21, 8-5, 8-6
Librarian, tape	9-13
Libraries of main programs	19-1
Libraries of subprograms	18-3*
Libraries, system	18-1*
"LIBRARY	8-8*, F-1
Library	9-3, 13-16, 14-12, 16-1, 17-9
Library of binary routines	17-1*
LIBRARY (EDITLIB)	17-3*
Library, condense	17-7*
LIMERR	13-21, 13-22*

## L (continued)

"LIMIT	5-8*
Limitations (Intercom)	4-6
Limited-distribution	7-1
Limit, time, storage, message	15-7
Line feed	4-5, 4-29
Line length (Intercom)	4-20
\$Line limit (PL)	13-32
Line limit (PL)	8-4
Line terminator	9-2
Linear	18-4
Linear algebra	18-4, 18-6
Linear equations	18-6
Lines per inch (printer)	C-2*
LINPACK	18-6*
LINWOOD	19-4*
List	2-2
List option	13-4
List (NETED)	4-7
LISTBIN	15-8
\$Listing options (LO)	14-3
LISTLIB (EDITLIB)	17-4*
"LISTMF	11-14, F-1
List-directed	13-7, 13-35
List, burstable	13-2
Literals	14-13
\$LO	13-4, 14-3
"LOAD	8-1, 8-4
Load	4-21, F-2
Load map	8-8, 13-16, 13-28
Loader	8-1*, 15-2, 15-5, F-2
Loader errors	4-5, 15-2
Loading, automatic	17-1
Loading, static	13-33
LOADLDR	19-9
"LOADPF	9-13, 9-15, F-1
Local	8-8
Local file functions (lf)	18-6
Location, transfer by	13-27
@LOCK	4-17
@LOCK,OFF	4-20*
@LOCK,ON	4-16, 4-20*
Logarithmic plot	18-13
Logical	D-3
Logical record	3-2, B-1, g-2*
LOGIN	4-2*, 4-4, 4-5, 4-6

## L (continued)

LOGOUT	4-4*
Lost time	2-1
Lower case	4-27*
Lowest cycle (LC)	9-7
L-tape	11-11, 11-13

## M

@M (MESSAGE)	4-17*
Machine tools	20-3
Machine-dependent (MD)	13-4
Main program	8-13
Mainframe	1-2, 1-3, 1-4, g-1
Major key (IS)	14-7
Manual	6-5
"MAP	8-8, F-1
MAP (LDSET)	8-6
Map, cross reference	13-33, 14-12, 15-5
Map, load	8-1, 8-8, 15-5
Mass storage limit	5-8, 15-7
Mass Storage System (MSS)	1-1, 1-4, 3-2, 7-4, 10-1*, 20-28
Mass storage, rotating	9-1
Master pack	9-13
Math library	13-21
Math Science Library	18-6*
Matrix	18-6
\$MD	13-4
Medium-speed terminal	3-3, 21-2, B-1
Memory	3-2, 4-6, 5-4, g-2*
Memory management	13-33
Memory map	15-5
Memory preset	4-22
Memory, central (CM)	5-3*
"MERGE	20-18
Merge	12-3, 16-1, 20-15
Message limit	15-7
Message (Intercom)	4-16
Message, dayfile	13-28
Message, error	13-29
Message, error (remote batch)	3-7
MFE	1-3, 5-3
MFF	1-4, 5-3
MFG	1-4
Microfiche	5-15, 23-1*

## M (continued)

Microfiche, classified	7-3*
Micro-to-mainframe file transfer	24-3*
MIMIC	20-9*
Minnesota Fortran (MNF)	13-1, 13-37*
MINPACK	18-6*
Missing subprograms	13-28
Mixed-mode	13-27
MNF (Minnesota Fortran)	13-1, 13-37*, F-2
MNSRDC	19-1*, 19-8
"MODE	5-8
Mode error	5-8, 13-21, 15-4, 15-5
Modify (MD)	9-7
"MOUNT	9-14, F-1
MOVLEV	13-28
MRL, maximum record length	5-11
"MSAUDIT	10-3*
MSAUDIT	6-3
"MSCHANG	10-4*
"MSFETCH	10-3*
MSFS communications error (MSS)	15-3
MSL	18-6*
"MSPASSW	10-4*
"MSPERMT	10-4*
"MSPURGE	10-3*
MSS	10-1*, 10-3
MSS errors	15-3
"MSSTORE	10-3*
MT/NT/GE/PE/HD count	5-9, 5-10
Multiple cycles	9-8
Multiple entry	13-28
Multiple read	9-8
Multiplexors	1-1*
Multireel file	11-9
Multi-file reel	11-14
Multi-read (MR)	9-7
@MYQ	4-13*, 4-14*

## N

NALCON	24-1
NAMELIST	13-35, 13-35
Name, transfer by	13-27
Narrow paper	5-15
NASTRAN	20-10*

## N (continued)

Navy Electronic Mail System (NEMS)	24-2
Navy Laboratories Computer Network	24-1
NCD (ROUTE)	5-14*
NEMS (Navy Electronic Mail System)	24-2
NETED	4-7*, 4-9, 4-11, 4-26, 15-8,
	21-5, 24-3, F-2
Network	1-5, 24-1
NEWPEN	23-3
NEWPL	16-1
"NOGO	4-21, 8-4
Noise brackets	11-3*, 11-11
Nonlinear equations	18-6
Nonlinear least squares	18-6
Non-existing file	9-8
Non-linear	18-4, 20-1
Non-numeric (COBOL)	14-19
Non-standard Fortran	21-1
NORERUN	15-6*
NORING, tape	11-11, 11-13
Northwestern University	19-5, 20-9, 20-23
NOS/BE	1-5, 14-17
NOS/BE errors	15-5
"NOTE	5-8*
NOVACOM	19-4*
No-operation instruction	D-3
NSRDC (subroutine library)	13-23, 18-7*, 19-8, 19-9
NSRDC5 (subroutine library)	18-8*, 19-8, 19-9
NUCLEUS	8-8, 18-2*
Numeric data format	14-10
Numerical analysis	20-12
NUMERR	13-21, 13-22*

## O

Object file	13-29
Object modules	8-1
Octal	13-11, D-1
Octal-decimal conversion	E-1
Odd parity	12-1
@OFF (remote batch)	3-4, 3-5
Official	5-6
OFLREQ	23-2
\$OL	13-31
OLDLIB	12-2

## O (continued)

OLDPL	16-1, 19-3
OMNITAB	20-12*, F-2
@ON (remote batch)	3-4, 3-5
On-line	9-2
On-line storage	10-1
OPENMS	13-28
Operating system	1-5
\$OPT	13-5, 13-32
\$Optimization (OPT)	13-5, 13-32
Options	4-21
Option, LDSET	8-6
\$OPT=0 (Fortran)	13-27
ORG	14-6
OTHER (Documentation)	19-8
OUTPUT	5-6, 5-13, 13-35
Output file	14-14
Output queue	3-2*
Overlay	8-2, 8-7, 8-13*, 8-14, 8-19, 9-1, 13-28, 14-22, 18-9, 18-12
Overlay capsules	8-3, 8-16*, 8-18, 8-20
Overlay capsules (Fortran)	8-16
Overlay file name	17-9
Overlay in a library	17-9*
Overlay (COBOL)	14-22
Overlay (Fortran)	8-13
Overlay, absolute	17-9
Override file name	4-22
Override, files	8-4
Over-index blank common	13-29
OVLNAME	17-9

## P

\$P	13-32, 14-12
Packs	9-2
@PAGE	4-11, 4-23*
Page numbering	13-5
\$Page size (PS)	13-5, 13-32
\$Page width (PW)	13-5, 13-32
Paper tape	20-3
Paper tape (Intercom)	4-29*
Parameter	5-2*
Parameter substitution	6-1*

## P (continued)

Parameter (FTN5)	13-15, 13-38
Parameters, UPDATE	16-1
Parity	11-2, 11-6
Parity, tape	15-2
PASCAL	20-12*
Password	7-2, 7-6, 9-13
Password (Intercom)	4-2
Password (MSS)	10-1, 10-5
Password (PW)	9-7
Password, device set	7-6*
"PAUSE	9-14, F-1
PB	9-13
PC (personal computer)	24-3*
SPD	13-5, 13-32, 14-3, 14-12
PE	11-11, 11-13
Percent	4-5
Perforation skipping	C-2*
Peripheral processors (PP)	1-1*
Permanent error (MSS)	15-3
Permanent file	1-4, 9-10
Permanent file functions (PF)	18-7
Permanent file return codes	9-9
Permanent files	1-5, 3-2, 4-18, 9-3, 9-4, 9-13
Permanent files, classified	7-4*
Permission	10-5
Personal computer	24-3*
Personal data	5-6, 7-2
PF	13-23
PFRSTOR	9-4
Phase encoded	11-11, 11-13, 13-26
Phosphor, screen	22-2
Physical record unit (PRU)	9-2
Pipe stress	19-4
SPL	8-4, 13-5, 13-5*, 13-8*, 13-32, 13-32
PLFILE	18-17
"PLI	F-2
PLILIB	18-2*
Plot	18-10, 18-12, 18-17
PLOT 10	22-3, 22-6
PLOT 10 EASY GRAPHING	22-7, 22-8
PLOTPR	18-10
Plots, printer	18-10
Plotters, Calcomp	18-10, 18-13



## P (continued)

Plotters, off-line	18-10
Plotting	18-7, 18-17, 23-3
Plotting libraries, list	18-3
PL/I	20-12*
PM	5-12, 21-2
PMD	13-16
\$PMd (Post Mortem Dump)	13-32
\$PN	13-5
Pointer (NETED)	4-7
Polar plot	18-13
Policy	3-2, 5-4, g-2*
Polynomial	18-6, 18-13
Polynomial roots	18-7
POP936	18-17, 18-19
Port	4-2*
Position	12-2
Positional parameters	5-2*
Post Mortem Dump	13-9*, 13-16
\$Post Mortem Dump (PMD)	13-3, 13-32
Post-processor	18-17, 18-19
Precision	D-1
Preset memory	4-22, D-3
PRESET, PRESETA	8-6
Pre-compiler	13-39
Print density control	C-2*
\$Print density (PD)	13-5, 13-32, 14-3
Print forms	5-15
\$Print limit (PL)	8-4, 13-5*, 13-32
Print limit (PL)	13-8*
Print requirements	21-5*
Printer	13-29, 23-4
Printer carriage control	C-1*
Printer plots	18-10
Printer, send output to	4-12
Printer, user-programmable	C-3*
@PRIOR (remote batch)	3-6
Priority	3-2, 3-8, 5-4*
Priority groups	5-4
Privacy Act of 1974	5-6, 7-2*
Private	7-2, 10-5
"PRNTSPY	5-8*, F-1
PROCEDURE DIVISION (COBOL)	14-15
Procedure file	6-2
Procedures	6-1*, 9-3
PROCFIL	6-2*, 19-8

## P (continued)

Program documentation	19-8
PROGRAM statement (Fortran)	8-4, 13-7*, 13-34
PROSE	20-32*
Protocol, XMODEM	24-3*
PRU	9-2*, 11-1*, 11-5
PRUDMP	15-8
SPS	13-5, 13-32
Public	9-7, 10-5, 18-3
Public Law 93-579	7-2*
Public-access password	9-13
*PULLMOD	16-3*
Punch	2-2, 4-6, 5-13
PUNCHB	13-2, 13-30, 14-11
Punched cards	2-3, 5-15
PUNTAP	20-3
*PURDECK	16-3*
"PURGE	4-18, 9-6, 9-10, F-1
*PURGE	16-3*
Purge	9-3, 9-4
\$PW	13-5, 13-32
P0 (block time)	3-8

## Q

\$Q	13-33
@Q	4-13, 4-15*
Q (carriage control)	C-2*
\$QC	13-5
Query Update (QU)	20-8*, F-2
Queue	3-2, 3-8
Queue device	5-8, 5-12
Queue resident files	3-6
Queue, input	5-10
Quick compile (QC)	13-5
Quotes (COBOL)	14-13, 14-15

## R

\$R	13-33
R (carriage control)	C-2*
R (tape label)	11-12*
Random	11-1, 12-1, 16-2, 17-4, 19-3
Random access	9-1, 9-2, 13-28
Random numbers	18-5

## R (continued)

Range error	13-28
RANTOSEQ (EDITLIB)	17-4*
RATFOR	13-1, F-2
Rational Fortran (RATFOR)	13-39*
RB, record blocking	5-11
*READ	16-3*
Read	13-7
@READ (remote batch)	3-5
READMS	18-9
READY..	4-23
Read-only (XR)	9-7
READ, formatted (FTN)	18-9
READ, unformatted (FTN)	18-9
Record	21-5
Record blocks (RB)	9-2
Record identification cards, COPYN	12-3
Record level	12-7
Record Manager (RM)	8-6, 9-1, 9-16, 13-29, 14-16, 14-17, 15-2
"RECOVER	9-15, F-1
RECOVR	13-21, 13-22*
"REDUCE	8-9*
Reference address (RA)	g-2*
Refund	2-1, 19-11
Registers	1-1*
Regression analysis	18-5, 19-4, 19-5, 20-23
Reload	9-15
Relocatable	4-22, 17-1
REMARK	21-4
Remote batch	3-3, 4-6
Remove job from queue (Intercom)	4-12*
RENAMAC	6-6, 9-4
"RENAME	9-4, 9-6, 9-11, F-1
@REP (remote batch reprint)	3-4
REP (ROUTE)	5-14*
REPLACE (EDITLIB)	17-3*
Report generation	20-13
Reprieve	13-29, 15-7
\$Reprieve (ER)	13-3, 13-21, 13-31
Reproduce	2-2
"REQUEST	5-8*, 9-6, 9-10, 9-14, 11-13, F-1
Request	11-1, 11-3, 13-23
Rerun	15-6*

## R (continued)

Reset	22-2
Resources	3-2
"RESTART	20-4*, F-1
Restart	20-6
*RESTORE	16-3*
Resume	4-4*
"RETAIN	5-8*, F-1
Retention (RP)	9-7, 9-13
Retention, files	9-4
Retention, tape	11-12*
Retrieval	20-13
"RETURN	4-6, 4-8, 4-18, 5-9*, F-1
Return	13-14
Return codes, permanent file	9-9
Returns, non-standard	13-16
"REVERT	6-1*, F-1
Revision	iii
\$REW	13-6
@REW (remote batch)	3-4, 3-4
"REWALL	5-9*
"REWIND	5-9*, 8-4, F-1
*REWIND	16-3*
Rewind	13-6
REWIND (EDITLIB)	17-3*
"RFL	3-2, 8-9*, F-2
Ring, tape	11-11, 11-13
RM error	15-2
RNF	20-32*
Root segment	8-10
Roots, polynomial	18-7
\$ROUND	13-6, 13-33
"ROUTE	4-11, 4-12, 4-28, 5-6, 5-12*, 5-16, B-1, F-1
Route	2-2, 13-23
RP, device set	5-4
RT, record type	5-11
RUN Fortran	13-1, F-2
RUN (EDITOR)	4-9
RUNOFF	20-32
RUN2P3	18-2*
Run, compiler	4-6
RVD, tape	15-2

AD-A150 162 COMPUTER CENTER CDC REFERENCE MANUAL(U) DAVID W TAYLOR 5/5  
NAVAL SHIP RESEARCH AND DEVELOPMENT CENTER BET..  
D Y SOMMER ET AL. SEP 84 DTNSRDC/CMLD-84-10

AD-A150 162 COMPUTER CENTER CDC REFERENCE MANUAL(U) DAVID W TAYLOR 5/5  
NAVAL SHIP RESEARCH AND DEVELOPMENT CENTER BET..  
D Y SOMMER ET AL. SEP 84 DTNSRDC/CMLD-84-10

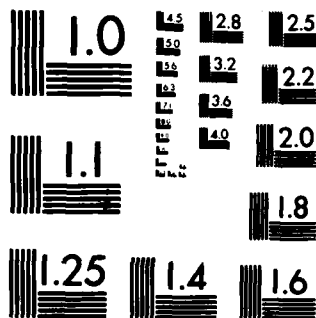
AD-A150 162 COMPUTER CENTER CDC REFERENCE MANUAL(U) DAVID W TAYLOR 5/5  
NAVAL SHIP RESEARCH AND DEVELOPMENT CENTER BET..  
D Y SOMMER ET AL. SEP 84 DTNSRDC/CMLD-84-10

UNCLASSIFIED F/G 9/2 NL

UNCLASSIFIED F/G 9/2 NL

UNCLASSIFIED F/G 9/2 NL

[illegible][illegible][illegible]



MICROCOPY RESOLUTION TEST CHART  
NATIONAL BUREAU OF STANDARDS-1963-A

## S

S (carriage control)	C-2*
SANDIA	18-8*
"SATISFY	8-5
SAVE (FTN5)	13-15
SAVE (NETED)	4-7, 4-18*
SAVE (SEGLOAD)	8-10
SC (ROUTE)	5-14*
SCCALC	18-15*
Scientific libraries, list	18-3
Scientific Subroutine Package (IBM)	18-9*
Scope 3.3	A-2
Scratch	11-1
Scratch files	5-8, 9-1
@SCREEN	4-20*
Screen	22-2, 22-5
SC4020, convert	18-15*
Search (PAGE)	4-25
Secret	5-6
Sector	21-5
@SECURE	4-4*
Security	4-3, 7-1*
Security (MSS)	10-1
Seek	9-2
"SEGLOAD	8-10*, 8-19
Segmentation	8-2, 8-10*, 8-11, 14-22, 18-9
SELECT (COBOL)	14-14
Semicolon	A-2
Semi-private	10-5
@SEND	4-17*
Send	2-1, 4-17
Send output to printer	4-12
\$SEQ (sequenced format)	13-6, 13-33
SEQTORAN (EDITLIB)	12-1, 17-4*
*SEQUENCE	16-3*
\$Sequenced format (SEQ)	13-6, 13-33
Sequential	9-1, 9-2, 9-17, 13-7, 16-2, 17-4
Sequential files	12-1
"SET	4-26*
SETAL (EDITLIB)	17-3*, 17-10*
"SETNAME	9-14
Share	19-7*
Shared files	14-5
SHARP	20-13*
Shorten a file	9-12

## S (continued)

SHORTIO	18-9*
SI tape	11-1*, 11-2, 11-5, 11-11, 11-13, 11-14, 18-12
Sign-on	4-2*
SIMII5	20-14*, F-2
Simple load	8-1
Simscrip	20-14*
Simulation	20-9
Simultaneous equations	18-5
SIS	9-16, 9-17, 12-1
"SISTAT	9-16, F-1
Site-id	5-4
@SITUATE	4-17, 4-20*
*SKIP	16-3*
Skip	14-7
"SKIPB	12-2*, E-1
SKIPB/SKIPF (EDITLIB)	17-3*
"SKIPF	12-2*, F-1
Skipping over perforation	C-2*
SKPFIL	13-23
"SLOAD	8-5
Slot tape	11-7*, 11-9
Slower execution	13-28
SNOBOL	20-14*, F-2
Sort	2-2, 14-14, 20-15, 20-18
Sorter	2-2
Sorting	18-7
"SORTMRG	20-15*
"SORT5	20-18*
SORT/MERGE	13-29, D-1, F-2
Source code	19-9
Source library, UPDATE	16-1
Spacing code (ROUTE)	5-14*
Special forms	5-15*, 21-2
Special functions	18-6, 18-7
Specification statements	13-27
SPLICE	11-8*
SPSS	19-8, 20-23
SPY	5-8, 13-23
SPYONF, SPYOFF	13-23
SRTLIB	18-2*
SSP	18-9*, 18-10
Stack	9-2
Statement function	13-11
Static	13-16, 13-33



## S (continued)

Statistical analysis	18-5
Statistical programs	19-5
Statistics	20-23
Status phone	4-2
Sticker label	11-9
Stop	13-12
Storage System, Mass (MSS)	10-1
Storage, tape	11-7
@STORE	4-8, 4-18*
Store (MSS)	10-3
Stranger tape	9-1, 11-2*, 11-5
Stress analysis	19-4, 20-1
String	13-11
String manipulation	20-14
String (PAGE)	4-25
Structural analysis	20-1, 20-10
Structure	13-39
Subprogram	13-14
Subprogram (COBOL) (SUB)	14-12, 14-13
Subprograms, libraries of	18-3*
Subscript	13-28, 14-19
\$Subscript bound (SB)	13-3
Subscripts (COBOL)	14-11, 14-16
\$SUB, SUBM	14-12
Suggestions	6-8
"SUMMARY	4-6, 5-9*
Summary	4-4
SUP (Intercom)	4-2
Swap	3-2
\$SY	14-3
SYMLIB	18-2*
Synchronous	22-2
Syntax check (QC)	13-5
\$Syntax check (SY)	14-3
"SYSBULL	4-4*, 5-9*, F-1
SYSIO	18-2*
SYSLIB	18-2*
SYSMISC	18-3*
SYSOVL	18-2*
System bulletin	4-4*, 5-9*
System libraries	18-1
System permanent files	9-7
System software	19-9
System 2000 (version 2.60)	20-30*
System 2000 (version 3.0)	20-25*

## S (continued)

SYSTEMC	13-21
System-logical-record	11-1, 11-2
S2K280	F-2
S2K30	20-25
S2000	20-25, F-2
S-tape	11-11, 11-13

## T

\$T	13-33
T (carriage control)	C-2*
TAPDMP9	15-8
Tape	3-2, 5-3*, 5-9, 5-9, 7-2, 9-1, 10-1, 11-1*, 11-9, 13-7,
	13-26, 15-2
Tape assignment	11-7
Tape character	A-1
Tape format	11-6
Tape label	11-11*
Tape librarian	9-13
Tape requirements	11-4*
Tape, binary	23-2
@TAPE,OFF	4-29*
@TAPE,ON	4-5, 4-29*
"TDUMP	15-8
@TEACH	4-6
Tektronix	4-27, 18-17, 22-2, 22-4, 22-6, 22-7, 22-8
Tektronix 4015	22-2*
Tektronix 4027	22-7*
Tektronix 4051	22-6*
Tektronix 4662	22-4*
TEK30	22-3, 22-4, 22-6
TEK300	18-17, 18-19, 22-3
TEK48	22-3
TEK480	18-17, 18-19, 22-3
Telephone	4-2
Telephone, computer access	1-3, 1-4
Teletype-compatible	1-5, 4-1
Terminal	13-7, 13-29
Terminal identification	5-12
Terminal, medium-speed	3-3
Termination, error	13-4
Terminator	5-1*

## T (continued)

Text processors	20-32*
Text replacement (NETED)	4-8
THREE-D	18-16*
TID (ROUTE)	5-12*
Time	3-8, 5-4, 14-4
Time limit	4-5, 4-6, 4-19, 15-7
Timeout error (MSS)	15-3
Time-series	19-4
Time-sharing	4-1
\$Time-sharing FTN (TS)	13-33, 13-36
Time, job card	5-3*
TODAYS-DATE	14-6
TRACE	13-33, 14-11, 15-5
\$Traceback (TB)	13-3
Traceback, error	13-33
Trailer, tape	11-14
TRANPK	9-15
"TRANSF	5-10*, F-1
Transfer	11-1, 11-6
Transfer by location	13-27
Transfer by name	13-27
Transfer by value	13-27
Transfer, file, micro-to-mainframe	24-3*
Translate	24-3
Translation	11-6
"TRANSPF	9-13, 9-15
Tree	8-10
Trip count	13-3
Trouble form	2-1
True	D-3
\$TS	13-33
@TURNKEY	4-3*
Turnkey password	7-1, 7-2
Turnkey (Intercom)	4-2
Turnkey (TK)	9-7
Type	13-11

## U

\$U	14-12
UN (MSS)	10-5*
Unblocked	9-1
Unclass	5-6, 7-5
Unclassified	9-3

## U (continued)

Unformatted	13-28
Unit record	1-5, 4-23, 9-2
UNIVAC	11-6
Universal password	9-13
Unlabelled	11-2
"UNLOAD	4-8, 5-9*, 5-10*, F-1
Unsatisfied externals	15-5
Unsatisfied reference	13-28
"UPDATE	16-1*, F-2
Update	9-3, 12-1, 14-9, 21-5
UPDATE directives	16-3*
User device set	7-1, 9-13
User ID	9-5
User initials	4-2, 5-3, 5-5, 10-5, g-2*
User library	17-1*
User name	4-2
User parameter (FTN5)	13-10*
User return codes	9-9
User Services	2-1
USER-ID	4-2, g-2*
User-programmable printer	C-2*
USE, USEP loader	8-7
USING (COBOL)	14-13, 14-15
Utilities	2-2, 9-16, 12-1, 13-22
Utility	6-6
	19-2*, 19-8, 19-9
Utility, device set	9-15

## V

V (carriage control)	C-3*
Value, transfer by	13-27
Variable	13-27, 13-28
VAX, DEC	3-3
VIM	19-7*
Visual reel number (VRNO)	11-12
Volume serial number	9-13, 11-9
"VSN	11-9*, F-1
VSN	9-13, 11-1, 11-7, 11-12, 11-13, 11-14
VT100	4-27*

## W

W (tape label)	11-12*
@WAIT (remote batch)	3-4, 3-5
"WARNING	5-6*, F-1
*WEOR	16-3*
Word length	D-1
WORKING-STORAGE (COBOL)	14-13
Write	13-7
Write errors (tape)	11-3*
WRITMS	13-35
WTSET	13-2
WW (forms code)	23-1

## X

@XEQ	4-8, 4-21*, 4-22
Xerox	6-7
Xerox 8700	5-15, 23-4
XH, XI, XL (forms codes)	5-15
"XMODEM	24-3*
X-OFF	4-29*

## Y

*YANK	16-3*
-------	-------

## Z

\$Z	14-12
\$Z parameter	14-17
Zero-byte terminated	14-17
ZZ... files	4-5
Z-type records	4-23

## 0-9

026	2-2, 3-3, 4-28, A-1, B-1
029	2-2, 3-3, 4-28, 5-16, A-1, B-1
1700	B-1
200-UT	A-2, B-1
2550	1-3, 1-4

4015, Tektronix	22-2*
4027, Tektronix	22-7*, 22-8
4051, Tektronix	22-6*
4662, Tektronix	22-4*
580 printer	5-14*
6C (SORTMRG)	20-15
6250	5-3, 11-2, 11-3, 11-4, 11-5, 11-11, 11-13
6/7/8/9 card (eoi)	5-1*
7-track tape	11-2*, 11-11
7/8/9 card (eor)	5-1*, 21-1, 21-5, g-2*
819 disk	9-2
844 disk	9-2
885 disk	1-5, 9-2
936 Calcomp	18-12*, 23-3
9-track tape	11-2*, 11-11, 12-13
spec	
-FZ	14-6
*/ comment	17-4*
\$ delimiter	5-2*, 8-12
%A	4-5, 4-7
%EOF	4-5
%EOR	4-5
%S	4-5

Initial Distribution

Copies:

12 Director  
Defense Technical Information Center (DTIC)  
Cameron Station  
Alexandria, Virginia 23314

Center Distribution

Copies:

1	18/1809	Gleissner, G. H.
1	1804	Avrunin, L.
1	1805	Cuthill, E. H.
2	1808	Wildy, D.
1	182	Camara, A. W.
1	184	Schot, J. W.
1	185	Corin, T.
1	187	Zubkoff, M. J.
1	189	Gray, G. R.
1	189.2	Hayden, H. P.
1	189.3	Morris, J.
150	1892.1	Strickland, J. D.
20	1892.2	Sommer, D. V.
1	1892.3	Minor, L. R.
1	1894	Seals, W.
1	1896	Glover, A.
1	1896.2	Dennis, L.
1	522	TIC (C)
1	522.2	TIC (A)
1	93	Patent Counsel

DTNSRDC ISSUES THREE TYPES OF REPORTS

1. DTNSRDC REPORTS, A FORMAL SERIES, CONTAIN INFORMATION OF PERMANENT TECHNICAL VALUE. THEY CARRY A CONSECUTIVE NUMERICAL IDENTIFICATION REGARDLESS OF THEIR CLASSIFICATION OR THE ORIGINATING DEPARTMENT.
2. DEPARTMENTAL REPORTS, A SEMIFORMAL SERIES, CONTAIN INFORMATION OF A PRELIMINARY, TEMPORARY, OR PROPRIETARY NATURE OR OF LIMITED INTEREST OR SIGNIFICANCE. THEY CARRY A DEPARTMENTAL ALPHANUMERICAL IDENTIFICATION.
3. TECHNICAL MEMORANDA, AN INFORMAL SERIES, CONTAIN TECHNICAL DOCUMENTATION OF LIMITED USE AND INTEREST. THEY ARE PRIMARILY WORKING PAPERS INTENDED FOR INTERNAL USE. THEY CARRY AN IDENTIFYING NUMBER WHICH INDICATES THEIR TYPE AND THE NUMERICAL CODE OF THE ORIGINATING DEPARTMENT. ANY DISTRIBUTION OUTSIDE DTNSRDC MUST BE APPROVED BY THE HEAD OF THE ORIGINATING DEPARTMENT ON A CASE-BY-CASE BASIS.



**END**

**FILMED**

**3-85**

**DTIC**